UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/748,503	12/26/2000	Cheryl Kellond	APP1P003/44379/03331	3355

7590 10/14/2004

STEVE GUPTA
VICE PRESIDENT FINANCE APPAREON
1100 ISLAND DRIVE
REDWOOD CITY, CA 94065

EXAMINER	
JARRETT, SCOTT L	

ART UNIT	PAPER NUMBER
3623	

DATE MAILED: 10/14/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/748,503	KELLOND ET AL.	
	Examiner	Art Unit	
	Scott L. Jarrett	3623	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☐ Responsive to communication(s) filed on _____.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) 1-20 is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) 1-20 is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☒ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

Drawings

1. The drawings are objected to because Figures 1, 11-19 are illegible. Corrected drawing sheets in compliance with 37 CFR 1.121(d) are required in reply to the Office action to avoid abandonment of the application. Any amended replacement drawing sheet should include all of the figures appearing on the immediate prior version of the sheet, even if only one figure is being amended. The figure or figure number of an amended drawing should not be labeled as "amended." If a drawing figure is to be canceled, the appropriate figure must be removed from the replacement sheet, and where necessary, the remaining figures must be renumbered and appropriate changes made to the brief description of the several views of the drawings for consistency. Additional replacement sheets may be necessary to show the renumbering of the remaining figures. The replacement sheet(s) should be labeled "Replacement Sheet" in the page header (as per 37 CFR 1.84(c)) so as not to obstruct any portion of the drawing figures. If the changes are not accepted by the examiner, the applicant will be notified and informed of any required corrective action in the next Office action. The objection to the drawings will not be held in abeyance.

Claim Objections

2. Claim 6 is objected to because of the following informalities: the dependent claim is self-referential. The examiner read the claim as being a dependent of Claim 1 and not Claim 6 as claimed. Appropriate correction is required.

Claim Rejections - 35 USC § 112

1. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

2. Claims 1-20 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite and failing to point out and distinctly claim the subject matter which the applicant regards as the invention.

Regarding Claims 1, 8 and 15 the disclosure does not clearly define the phrase "template" when referencing either project or process templates. A template as claimed can comprise a plurality of components or items including, but not limited to: computer code, objects, classes, business rules, logic, system interfaces, document formatting guidelines or any combination thereof thereby making the term "template" as claimed vague and indefinite. The examiner read template to mean any available software component comprising any of the items discussed above.

Regarding Claims 1, 7, 8 and 15 the disclosure does not clearly define the phrase "module". The phrase module as claimed can be interpreted in a plurality of ways including but not limited to: a standardized system or construction that is designed for easy assembly or flexible use, a self-contained assembly of electronic components and circuitry, such as a stage in a computer, that is installed as a unit or a portion of a program that carries out a specific function and may be used alone or combined with

other modules of the same program. The examiner read module to mean a component of a software program.

Regarding Claim 1(e) the disclosure does not clearly define the phrase "plugging" when referencing "plugging" the dynamic supply chain module into a supply chain. The phrase plugging as claimed could encompass a plurality of definitions including but not limited to: the act of providing a computer system (hardware and software) with electrical power or instantiating supply chain modules in computer program thereby making the term "plugging" as claimed vague and indefinite. The examiner read plugging to mean the inclusion of a dynamic supply chain module (component) into a software program.

Regarding Claim 15 the disclosure does not clearly define the phrase "system." A system as claimed could contain a plurality of elements and without further definition of the system elements the phrase as claimed vague and indefinite.

Further regarding Claim 15 the disclosure does not clearly define the phrase "logic." Logic has a plurality of meanings including logic conditions embodied in software or a mode of reasoning thereby making the term "logic" as claimed vague and indefinite. The examiner read logic to mean logic conditions embodied in software.

Claim Rejections - 35 USC § 101

3. Claims 1-20 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

The basis of this rejection is set forth in a two-prong test of:

(1) whether the invention is within the technological arts; and

(2) whether the invention produces a useful, concrete, and tangible result.

For a claimed invention to be statutory, the claimed invention must be within the technological arts. Mere ideas in the abstract (i.e., abstract idea, law of nature, natural phenomena) that do not apply, involve, use, or advance the technological arts fail to promote the "progress of science and the useful arts" (i.e., the physical sciences as opposed to social sciences, for example) and therefore are found to be non-statutory subject matter. For a process claim to pass muster, the recited process must somehow apply, involve, use, or advance the technological arts.

Regarding Claims 1-7, claims 1-7 only recite an abstract idea. The recited method for providing a dynamic supply chain module in a supply chain of a plurality of businesses does not apply, involve, or use the technological arts since all of the recited steps can be performed in the mind of the user or by use of a pencil and paper.

Additionally, for a claimed invention to be statutory, the claimed invention must produce a useful, concrete, and tangible result. The claimed invention, as a whole, is

not within the technological art as explained above claims 1-7 are deemed to be directed to non-statutory subject matter.

Software, programming, instructions or code not claimed as embodied in computer-readable media are descriptive material per se and are not statutory because they are not capable of causing functional change in a computer. When such descriptive material is recorded on some computer-readable medium it becomes structurally and functionally interrelated to the medium and will be statutory in most cases.

Furthermore, software, programming, instructions or code not claimed as being computer executable are not statutory because they are not capable of causing functional change in a computer. In contrast, when a claimed computer-readable medium encoded with a computer program defines structural and functional interrelationships between the computer and the program, and the computer is capable of executing the program, allowing the program's functionality to be realized, the program will be statutory.

Regarding Claims 8-14 do not utilize the proper computer program product format and effectively recite descriptive material (software) per se. Claims 8-14 are therefore deemed to be directed to non-statutory subject matter where there is no indication that the proposed software is recorded on computer-readable medium and/or capable of execution by a computer. Examiner suggests that the applicant incorporate

into Claims 8-14 language that the proposed software is recorded on computer-readable medium and capable of execution by a computer to overcome this rejection.

Regarding Claims 15-20, claims 15-20 do not utilize the proper computer program product format and effectively recite descriptive material (software) per se and are therefore deemed to be directed to non-statutory subject matter where there is no indication that the proposed software is recorded on computer-readable medium and/or capable of execution by a computer. Examiner suggests that the applicant incorporate into Claims 15-20 language that the proposed software is recorded on computer-readable medium and capable of execution by a computer to overcome this rejection.

Correction required. See MPEP § 2106 [R-2].

Claim Rejections - 35 USC § 102

4. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

5. Claims 1-20 are rejected under 35 U.S.C. 102(b) as being unpatentable by Microsoft's Site Server 3.0 Commerce Edition as evidenced by Master Card's International Clarus E-Procurement (white paper, August 1999), Microsoft Site Server Commerce Edition (product brochure, 1998), Implementing Pipeline Interfaces in

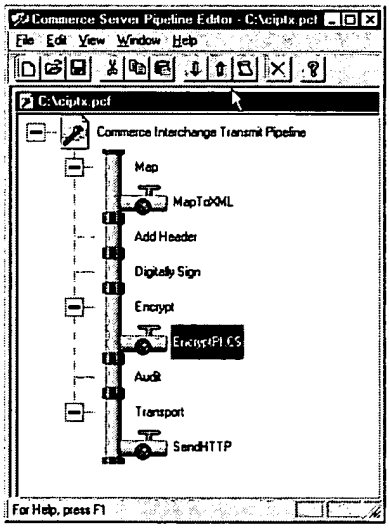
Microsoft Site Server 3.0: Converting Existing COM Components (June1999) and Marco Tabini Extending Site Server 3.0 Pipelines (article, Wrox Conferences, September 1999).

6. Regarding Claim 1 the Microsoft Site Server 3.0 Commerce Edition teaches a method, computer program product and system for providing a dynamic supply chain module in a supply chain of a plurality of businesses.

The Microsoft Site Server 3.0 Commerce Edition includes the Commerce Interchange Pipeline (CIP module) as the key piece of technology in delivering supplier integration solutions. A pipeline conceptually represents a business process decomposed into its essential processes, sub-processes, tasks and activities that are executed when the module is called as part of a dynamic supply chain (Master Card's International Clarus E-Procurement white paper; see at least Paragraph 1, Page 8; Figure 5, Page 14).

7. Per element 1(a) the Microsoft Site Server 3.0 Commerce Edition teaches a method, computer program product and system for providing a dynamic supply chain module in a supply chain of a plurality of businesses as discussed above and further comprising the step of selecting one or more project templates from a group of project templates to form a dynamic supply chain module, wherein each project template includes a plurality of process templates.

The Microsoft Site Server 3.0 Commerce Edition was shipped with a plurality of project and process templates. Once such template was the Transmit pipeline as shown below in the Microsoft Pipeline Editor. The Transmit pipeline template included a plurality of process templates (stages: Map, Add Header, Digitally Sign, Encrypt, Audit and Transport). Further each stage offered a choice of components (Master Card's International Clarus E-Procurement white paper; see at least Figure 5, Page 14).



8. Per element 1(b) the Microsoft Site Server 3.0 Commerce Edition teaches a method, computer program product and system for providing a dynamic supply chain module in a supply chain of a plurality of businesses as discussed above and further comprising the steps of manipulating the process templates to tailor the dynamic supply chain module.

The Microsoft Site Server 3.0 Commerce Edition was shipped with a plurality of project and process templates and a template editor. Once such template was the Transmit pipeline as shown above in the Microsoft Pipeline Editor. A simple right-mouse

click from within the pipeline editor adds a component at any stage thereby tailoring the pipeline module (Master Card's International Clarus E-Procurement white paper; see at least Figure 5, Page 14).

9. Per element 1(c) the Microsoft Site Server 3.0 Commerce Edition teaches a method, computer program product and system for providing a dynamic supply chain module in a supply chain of a plurality of businesses as discussed above and further comprising the step of the association of a module with a particular user. The Commerce Interchange Pipeline enables users, whether those users represent other systems, a human user (shopping cart for example) or other businesses, to exchange information regarding the particular business process embodied. Therefore it is inherent that each module, pipeline or pipeline stage, must be associated with a user or actor in the system.

10. Per element 1(d) the Microsoft Site Server 3. 0 Commerce Edition teaches a method, computer program product and system for providing a dynamic supply chain module in a supply chain of a plurality of businesses as discussed above and further comprising the step of choosing services, which acquire information from the user utilizing a network to further, tailor the dynamic supply chain module.

Microsoft Site Server 3. 0 Commerce Edition further teaches that pipeline components are commonly developed utilizing Microsoft Common Object Model (COM) technology. The Microsoft COM technology provides a framework for integrating

modules (components). This framework supports interoperability and reusability of networked modules (distributed objects) thereby enabling businesses to build systems by assembling reusable modules from different vendors, which communicate, via COM. Further COM provides a range of services for component interaction that the user can choose (utilize) as necessary in acquiring information from users on the network (Implementing Pipeline Interfaces in Microsoft Site Server 3.0: Converting Existing COM Components; see at least Paragraphs 1-3, Page 1).

11. Per element 1(e) the Microsoft Site Server 3. 0 Commerce Edition teaches a method, computer program product and system for providing a dynamic supply chain module in a supply chain of a plurality of businesses as discussed above and further comprising of the step of plugging the tailored dynamic supply chain module into a supply chain system as evidenced by the TransmitViaCIP module utilized as part of the Master Card electronic procurement solution (Master Card's International Clarus E-Procurement white paper; Page 14).

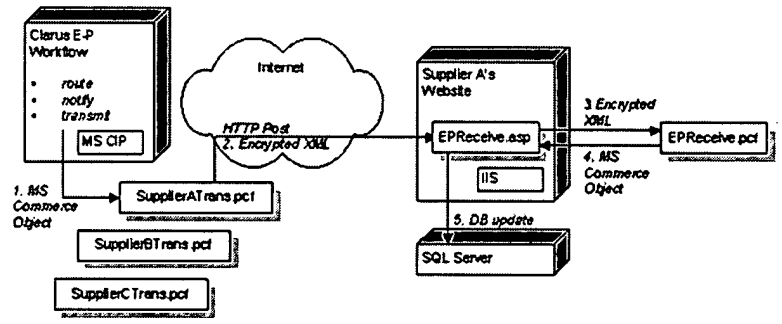
12. Regarding Claims 8 and 15, claims 8 and 15 recite similar limitations to Claim 1 and are therefore rejected using the same art and rationale as applied in the rejection of Claim 1.

13. Regarding Claim 3 the Microsoft Site Server 3.0 Commerce Edition teaches a method, computer program product and system for providing a dynamic supply chain

module in a supply chain of a plurality of businesses wherein the use of the network includes the Internet (Web-based) as evidenced by the Master Card's International Clarus E-Procurement application (e-procurement solution; Master Card's International Clarus E-Procurement white paper; see at least Page 1, Paragraph 4; Figure 4, Page 13; Figure 7, Page 19).

Microsoft Site Server 3.0 Commerce Edition further teaches that the product was built on Internet technologies including but not limited to Internet Information Server (IIS), Hyper Text Transport Protocol (HTTP), and eXtended Markup Language (XML) as shown below (Master Card's International Clarus E-Procurement white paper; Page 1, Paragraph 4).

Figure 4. Data Exchange Via CIP



14. Regarding Claims 10 and 17, claims 10 and 17 recite similar limitations to Claim 3 and are therefore rejected using the same art and rationale as applied in the rejection of Claim 3.

15. Regarding Claim 4 the Microsoft Site Server 3.0 Commerce Edition teaches a method, computer program product and system for providing a dynamic supply chain

module in a supply chain of a plurality of businesses as discussed above and further comprising selecting a plurality of users to interface with the dynamic supply chain module as evidenced in the Master Card's International Clarus E-Procurement implementation plan which called for the electronic procurement application to be deployed initially within a small pilot group, with incremental rollout to the entire user community (Master Card's International Clarus E-Procurement white paper; see at least Paragraph 5, Page 5).

16. Regarding Claims 11 and 18, claims 11 and 18 recite similar limitations to Claim 4 and are therefore rejected using the same art and rationale as applied in the rejection of Claim 4.

17. Regarding Claim 5 the Microsoft Site Server 3.0 Commerce Edition teaches a method, computer program product and system for providing a dynamic supply chain module in a supply chain of a plurality of businesses as discussed above wherein the project template allows the business to engage activities utilizing the network wherein the activities include a plurality of steps.

Microsoft Site Server 3.0 Commerce Edition further teaches that a pipeline conceptually represents a business process decomposed into its essential processes, sub-processes, tasks and activities that are executed when the module is called as part of a dynamic supply chain as discussed above. Microsoft Site Server 3.0 Commerce

Edition teaches that product was built on network (Internet) technologies as discussed above.

18. Regarding Claims 12 and 19, claims 12 and 19 recite similar limitations to Claim 5 and are therefore rejected using the same art and rationale as applied in the rejection of Claim 5.

19. Regarding Claim 6 the Microsoft Site Server 3.0 Commerce Edition teaches a method, computer program product and system for providing a dynamic supply chain module in a supply chain of a plurality of businesses as discussed above wherein the completion of the steps are tracked in a document (logs). The Microsoft Site Server 3.0 Commerce Edition teaches a plurality of means for recording, tracking, reporting and analyzing the processes and activities encompassed in the dynamic supply chain. Specific technologies for acting on logs or similar transaction information include but are not limited to: InetMonitor, NT Performance Monitor, and Microsoft Transaction Server.

20. Regarding Claims 13 and 20, claims 13 and 20 recite similar limitations to Claim 6 and are therefore rejected using the same art and rationale as applied in the rejection of Claim 6.

21. Regarding Claim 7 the Microsoft Site Server 3.0 Commerce Edition teaches a method, computer program product and system for providing a dynamic supply chain

module in a supply chain of a plurality of businesses as discussed above wherein the dynamic supply chain module is used to update process components of the supply chain.

The Microsoft Site Server 3.0 Commerce Edition further teaches that Commerce Interchange Pipeline enables users to exchange information regarding the particular business process embodied. One of the information exchanges core purposes in the updating (modification) of the systems, processes and their related data as a critical element for facilitation of the dynamic supply chain (Master Card's International Clarus E-Procurement white paper; see at least Paragraph 1, Page 8).

Claim Rejections - 35 USC § 103

22. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

23. Claims 2, 9 and 16 are rejected under 35 U.S.C. 103(a) as being unpatentable over Microsoft's Site Server 3.0 Commerce Edition as evidenced by Master Card's International Clarus E-Procurement (white paper, August 1999), Microsoft Site Server Commerce Edition (product brochure, 1998), Implementing Pipeline Interfaces in Microsoft Site Server 3.0: Converting Existing COM Components (June1999) and Marco Tabini Extending Site Server 3.0 Pipelines (article, Wrox Conferences, September 1999), as applied to Claims 1, 8 and 15 above, in view of Paul Korzeniowski

et al. (Trading Exchanges Have the 'Big Mo,' But Users Should Proceed with Caution, SupplyChainBrain.com, June 2000).

24. Regarding Claim 2 the Microsoft Site Server 3.0 Commerce Edition teaches a method, computer program product and system for providing a dynamic supply chain module in a supply chain of a plurality of businesses as discussed above.

Microsoft Site Server 3.0 Commerce Edition does not expressly state providing the dynamic supply chain wherein the businesses are apparel businesses.

Korzeniowski et al. teaches providing the dynamic supply chain wherein the businesses are apparel businesses.

Korzeniowski et al. teaches the use of trade exchanges (market wide supply chains), such as i2's SoftGoodsMatrix.com Business-to-Business marketplace for the retail and apparel industries, utilized for example by the VF Corporation which will be connecting to its key suppliers (Paragraphs 5-8, Page 6). Korzeniowski et al. further teaches trade exchanges as the hottest trend in supply chain management (Paragraph 1, Page 1) enabling the participating businesses to gain a competitive advantage (Paragraph 3, Page 4).

It would have been obvious to one skilled in the art at the time of the invention to leverage Microsoft Site Server 3.0 Commerce Edition to provide the dynamic supply chain modules for a dynamic supply chain in the apparel industry in view of the

teachings of Korzeniowski et al. in order to reap the benefits of increased competitive advantage to be gained by transforming key business processes (Paragraph 3, Page 4).

25. Regarding Claims 9 and 16, claims 9 and 16 recite similar limitations to Claim 2 and are therefore rejected using the same art and rationale as applied in the rejection of Claim 2.

Conclusion

The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

- Notani et al. (U.S. Patent 6,397,191) teaches an object-oriented workflow for multi-enterprise collaboration where the workflow comprises objects associated with activities to be performed within the workflow.

- Lehmann et al. (U.S. Patent 5,737,727) teaches a process management system and method for graphically presenting a process further consisting of other sub-processes, work elements or tasks to be performed as part of the process.

- Lynn et al. (U.S. Patent 6,606,740) teaches a workflow processing framework providing common objects and business process for a workflow management system.

- Barr et al (U.S. Patent 5,182,705) teaches a computerized method and process for managing work.

- Microsoft Pipeline - David Blinn interview with DowJones Online teaches the use of Microsoft Site Server 3.0 Commerce Edition and in particular its associated

Commerce Interchange Pipeline functionality as a means for providing dynamic supply chain modules.

- Microsoft Site Server 3.0 – Product Brochure (December 1999) discloses the latest date the Microsoft Site Server 3.0 and its associated Commerce Interchange Pipeline functionality was available.

- World Wide Retail Exchange – Overview teaches the application of supply chain management tools being leveraged by the retail industry as early as March 2000.

- Delivering 24/7 Solutions for Members of the World Wide Retail Exchange an i2 Technologies, Inc. supply chain management tools being leveraged by the retail industry.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Scott L. Jarrett whose telephone number is (703) 305-0587. The examiner can normally be reached on 8:00AM - 5:00PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Hafiz Tariq can be reached on (703) 305-9643. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

SJ
10/7/2004

Susanna Diaz
SUSANNA M. DIAZ
PRIMARY EXAMINER
AU. 3623

Notice of References Cited	Application/Control No. 09/748,503	Applicant(s)/Patent Under Reexamination KELLOND ET AL.	
	Examiner Scott L. Jarrett	Art Unit 3623	Page 1 of 2

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
	A	US-5,737,727	04-1998	Lehmann et al.	705/7
	B	US-6,397,191	05-2002	Notani et al.	705/9
	C	US-6,606,740	08-2003	Lynn et al.	717/100
	D	US-6,278,977	08-2001	Agrawal et al.	705/7
	E	US-5,182,705	01-1993	Barr et al.	705/11
	F	US-			
	G	US-			
	H	US-			
	I	US-			
	J	US-			
	K	US-			
	L	US-			
	M	US-			

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N					
	O					
	P					
	Q					
	R					
	S					
	T					

NON-PATENT DOCUMENTS

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
	U	Mastercard's Implementation of Clarus E-Procurement by Clarus Coporation and Microsoft Corporation, August 1999 http://www.microsoft.com/siteserver/commerce/DeployAdmin/ResKit.asp, retireved October 5, 2004
	V	Microsoft Site Server Commerce Edition, Product Brochure Microsoft - 1998
	W	Trading Exchanges Have the 'Big Mo,' But Users Should Proceed With Caution, Global Logistics & Supply Chain Strategies, June, 2000, http://www.supplychainbrain.com/archives/6.00.TradeExchange.htm?adcode=10, retrieved October 5, 2004
	X	Head to Head at your service - David Blinn interview with Dow Jones Online - July 1998; http://www.dnjonline.com/articles/head2head/iss7_head2head.html, retrieved October 5, 2004

*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.

Notice of References Cited	Application/Control No. 09/748,503	Applicant(s)/Patent Under Reexamination KELLOND ET AL.	
	Examiner Scott L. Jarrett	Art Unit 3623	Page 2 of 2

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
	A	US-			
	B	US-			
	C	US-			
	D	US-			
	E	US-			
	F	US-			
	G	US-			
	H	US-			
	I	US-			
	J	US-			
	K	US-			
	L	US-			
	M	US-			

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N					
	O					
	P					
	Q					
	R					
	S					
	T					

NON-PATENT DOCUMENTS

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
	U	World Wide Retail Exchange - Overview http://www.worldwideretailexchange.org/cs/en/about_wwre/overview.htm, retrieved october 5, 2004
	V	Delivering 24/7 Solutions for Members of the World Wide Retail Exchange i2 Technologies, Inc. Whitepaper, http://www.i2.com/customers/softgoods.cfm, retrieved October 5, 2004
	W	Marco Tabini Extending Site Server 3.0 Pipelines Wrox Conferences, September 1999 http://www.topxml.com/conference/wrox/1999_dc/text/marcopipe.asp, retrieved October 5, 2004
	X	Implementing Pipeline Interfaces in Microsoft Site Server 3.0: Converting Existing COM Components June1999 http://www.microsoft.com/siteserver/commerce/DeployAdmin/ResKit.asp, retireved October 5, 2004

*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.

Master Card International's Implementation of Clarus E-Procurement

CLARUS
World Class Applications...Breakthrough in Time®

WWTS Deployment Assets digital nervous system Microsoft TechNet

By Clarus Corporation and Microsoft Corporation

Version: Release 1.0

MasterCard

Company Profile

Since its inception 30 years ago, MasterCard International has grown from a US-based credit card system into a leader in the multi-trillion dollar global payments industry, with credit, debit and chip cards as core competencies. In 1997, 23,000 member financial institutions recorded 6.5 billion transactions worth $602 billion through MasterCard.

Solution Overview

This global payments industry leader is rolling out an advanced corporate purchasing solution based on Microsoft® BackOffice® technologies: Clarus™ E-Procurement from Clarus Corporation, a software vendor and Microsoft Certified Solution Provider. The sophisticated system has slashed the average time required to fill a purchase order by 70% and cut the cost of processing purchase orders from $125 to $40, even as it delivers valuable reporting data.

Situation

MasterCard's corporate procurement process was paper-based and labor-intensive. Looking to cut costs and improve delivery times, while simultaneously enabling buyers to take a more strategic approach to their jobs, the company decided to automate and streamline its procurement process by bringing it online.

Business Solution

Working with Clarus, MasterCard deployed a cutting-edge, Web-based electronic procurement solution powered by Clarus E-Procurement, Microsoft Internet Explorer, and Microsoft BackOffice technologies. The system gives users easy access to a cross-supplier product catalog over the corporate intranet. When the user makes a selection, the system automatically creates requisitions, routes them for approval, and submits them electronically to the relevant supplier. Used now by 750 people, the solution will be rolled out in a controlled manner to 2,300 users across MasterCard's entire enterprise.

Benefits

Clarus E-Procurement has empowered MasterCard to slash the average time required to fill a purchase order by 70%—down from four days to one—and to cut the cost of processing purchase orders from $125 to $40—a 68% reduction. And there are bigger savings to come: the company was already saving an estimated $10,000 per month on office supplies, computer hardware, and computer software alone during an early pilot phase with just 250 users. Meanwhile, the system is giving MasterCard valuable reporting data enabling it to get maximum value for every dollar it spends.

Project Overview

Vision Statement

To provide MasterCard employees with one, easy-to-use system for ordering non-production, maintenance, repair and operations (MRO) goods and services across all approved suppliers that exceeds employee service expectations while ensuring compliance with contracted suppliers and improving business resource management.

Project Scope

Like most large-scale projects, the implementation of Clarus E-Procurement at MasterCard required a phased approach.

Phase 1: Pilot Program

A pilot program to prove the concept and the value of the proposed solution took place over a five-week period from project planning through rollout.

The initial implementation had to be a fully functional, standalone system. Integration with other systems was not desirable during this phase of the project as they were in the process of being re-engineered.

This phase of the project included carefully selected requisitioners, buyers and administrators from both the St. Louis and New York sites. The primary focus of the pilot was on office- and computer-related supplies in addition to promotional goods for marketing events.

Activity	Effort	Timeline
Project Planning	1 Day	Week 1
Infrastructure Enablement	3 Weeks	Week 2-4
Administrator Training	2 Days	Week 2
Supplier Enablement	3 Days	Week 3-4
Software Installation & System Test	2 Days	Week 4
Database Configuration	5 Days	Week 4
User Training & Rollout	2 Days of 30-minute training sessions in two locations	Week 4-5
Feedback	Ongoing	Week 2+

The pilot program continued for several months, first using a small base of users to qualify the product's ability to perform, then qualifying the product from a user-acceptance standpoint.

Phase 2: Production Rollout

The first version of Clarus E-Procurement was rolled out at MasterCard's New York and St. Louis sites starting in August 1997. These users comprised the majority of administrative personnel and the entire purchasing department. The narrow scope of purchases was also expanded to include all externally procured goods and services.

Over the ensuing eight-month period, MasterCard conducted user orientation sessions and provided support through its Information Technology and Help Desk groups to smooth the transition of 750 users.

Phase 3: Enterprise Rollout

The next phase of the project was subject to the addition of feature enhancements to Clarus E-Procurement. These included the ability to:

- Handle virtually all procurement activities, from internal services to complex marketing requests
- Take advantage of Microsoft Commerce Interchange Pipeline (CIP) to communicate with suppliers, SUPPLY CHAIN

Suggestions gathered from user feedback were also added to the product.

All requisite enhancements have been added and the product is currently being rolled-out to all MasterCard employees. This phase of the project is scheduled for completion within the first half of 1999.

Phase 4: Back-end Integration

Clarus E-Procurement will be integrated with MasterCard's Enterprise Resource Planning (ERP) application—the installation of which is incomplete at this time—during this phase of the project.

Resource Requirements

Pilot Program

Initial deployment of Clarus E-Procurement involved a full-time project manager from MasterCard, with part-time support from various staff members. The project manager has been responsible for the project from mission through customer feedback.

Every successful project needs a champion, and this was no exception. This individual, an employee in the Purchasing organization, promoted solution adoption internally and mobilized the resources needed to deploy the system and accomplish the vision. This included selling the Information Technology (IT) department on the benefits of using the Microsoft technology platform as the infrastructure upon which to deploy the solution.

Additional support was required from key people in the IT, Purchasing, and Finance organizations.

Production Rollout

The production rollout required approximately three full-time people over an eight-week period to manage the project and provide connectivity for 750 users.

Support for Clarus E-Procurement beyond production rollout, including Catalog Management, Business Rules Management and User Profile Management, currently requires one half-time employee.

The person responsible for managing the entire system at MasterCard is a Purchasing employee. It's important to note that this individual is not required to have system programming experience.

The IT department's involvement has been limited to acceptance of the environment, installation of the hardware and network architecture, and support for the server and desktop environment. No database administrator (DBA) has been involved except to validate the system architecture. This has allowed the IT department to focus on other issues within the organization.

Organizational Process Requirements

Organizational change and end-user acceptance are typically the primary factors influencing new system deployment and success. However, unlike replacing an ERP system, implementing Clarus E-Procurement involves an entirely new breed of application. It was the first Web-based application at MasterCard, as it has been at many other Fortune 2000 companies.

Mostly casual users, rather than functional users, access this application at MasterCard. Therefore, end-user interaction and acceptance had to be handled differently than rollout of a replacement for an existing functional user application.

With this type of application, acceptance of process change is addressed by creating or modifying for end-users and purchasing professionals, and by communicating and monitoring compliance with those procedures. Additionally, IT requirements are identified and responsibilities assigned.

While this application provides significant benefits to the organization through process change and management information, it requires compliance by end-users to achieve results. Too many changes at once would thwart the effort and reduce the likelihood of success.

End-Users

The first step toward change is to provide end-users with access to the application. Rollout occurs in a controlled fashion while winning acceptance from key users. User-evangelists are very important to the success of a voluntary-use application. In addition, incentives to use the application must be in place, because individuals are not specifically measured on its use, as they would be with a functional use application.

As Clarus E-Procurement can be modified at any time without programming, no immediate changes to MasterCard's business rules and procedures were required. As a result, purchasing procedures were left unchanged for the first phase to allow validation of the system, ensure compliance, and reduce end-user hesitancy. Therefore, the only change a user had to endure was to use a new automated system instead of an old, manual, paper-based process. As might be imagined, this was a very welcome improvement.

With information never before available to the Purchasing organization, business rules and procedures can be modified and processes changed at any time as a positive result of the system. This change is a welcome one for those using

the old system.

Purchasing Professionals

The greatest change resulting from MasterCard's implementation of Clarus E-Procurement is to the purchasing professional. Though hired as valuable knowledge workers, these individuals have been burdened with extensive paperwork; routinely performing tedious data entry tasks, fulfilling end-users' information and status requests, handling complaints and manually expediting and following up on orders with suppliers.

Purchasing professionals now can work at a higher, more strategic level—fulfilling the requirement for which they were originally hired. They no longer have to manually enter paper requests into a purchasing system. This task is now performed at the frontline—by the end-user. Nor must they field questions about order status from requisitioners. The system proactively provides this information to the end-user via e-mail. This information is also available to the end-user through the Web-based application.

Purchasing professionals now use the system to obtain information that was unavailable with the old, manual, paper-based systems. Armed with this knowledge, they can better anticipate supplier opportunities and negotiate contracts that are more favorable for MasterCard. They can also provide management with much better and more timely information about the purchasing habits of the entire organization.

Because Clarus E-Procurement uses Microsoft SQL Server™, database administration is minimal and access to information is secure, yet readily available through Clarus E-Procurement's standard reports, as well as a range of widely available, third-party tools. MasterCard uses Microsoft Access as a reporting tool. With it, purchasing professionals gain insights into the purchasing habits of the organization and Accounts Payable ensures the timeliness of vendor payments.

IT Professionals

Clarus E-Procurement frees IT Professionals from the burden of system administration. This is a significant and welcome change.

Often, systems are ruled out due to the burden they place on the IT organization. Clarus E-Procurement follows Microsoft's "Zero Administration" philosophy by ensuring that the application only depends on its knowledge experts for maintenance. This reduces the burden on the IT organization to manage an application for which they are not experts.

IT has enjoyed benefits from the new system, too. Because MasterCard uses Microsoft Windows NT® Workstation as the standard desktop operating system, the best time to authorize an individual's use of an application is during initial system configuration. Therefore, the IT department now sets up the user and Microsoft Internet Explorer on the desktop every time a new machine is received. Desktop components used by the application and the user profile are verified upon installation.

Clarus developed a utility that enables administrators to push E-Procurement's components to the desktop for rapid installation on existing machines.

Functional Requirements

MasterCard had a number of specific objectives they wanted to accomplish. Before any electronic procurement system would be accepted, it had to:

- *Be scalable*

Be capable of supporting transactions generated by every employee in the organization.

- *Have a supplier integration strategy that met the company's core business requirements and objectives.*

  Lower the number of incomplete or incorrect orders that suppliers receive while simultaneously increasing overall order volume by reducing unauthorized buying.

- *Be easy to install, configure, maintain and use*

  Offer a "user-compelling" experience—so users prefer it to the old, manual paper-based systems. As a result, users will not subvert it by engaging in "maverick" buying practices. And it must do all this without placing an undue burden on the IT organization.

- *Integrate with ERP*

  Support seamless integration with an impending ERP implementation as well as existing back-end systems.

Through these objectives, MasterCard's Purchasing Department made "Customer Delight" their number one priority.

Scalability

A key requirement in MasterCard's electronic procurement strategy was to obtain a cost-effective, scalable architecture to meet its enterprise needs. By selecting Clarus E-Procurement, MasterCard made the strategic decision to closely align themselves with Microsoft's electronic commerce strategy. All Clarus products are created exclusively with native Microsoft technologies. As such, MasterCard's chosen solution is well suited to take advantage of the rapid advancements being made in Web-based technologies.

The MasterCard implementation plan called for the electronic procurement application to be deployed initially within a small pilot group, with incremental rollout to the entire user community.

When this type of application is implemented across an entire enterprise, user concurrency from 3% to 5% is expected. That is, between 3% and 5% of potential users are expected to access the system at any one time. Currently 750 users, mostly administrative personnel and purchasing professionals, have access to and actively use the system. Given their functional responsibilities, concurrency is significantly higher than 5%. In fact, these users generate about 60% of the total forecasted transaction volume for the enterprise.

The current system runs on a dual processor Pentium 300 server hosting the Microsoft SQL Server 6.5 database, Microsoft Internet Information Server 4.0 and Clarus Enterprise Application Server. As MasterCard's user base and transaction volume increase, the Clarus n-tier architecture will allow the application components to be distributed across multiple servers, thus providing much higher capacity. Based upon current benchmarks, the Clarus solution is expected to effectively scale to support a user base many times greater than MasterCard's future requirement.

Catalog Management

One of MasterCard's basic requirements for any electronic procurement solution was (past tense is used when referring to their other requirements... we should keep this consistent) an online, integrated product catalog. Their assumption was that errors created by the old, manual, paper-based processes would be dramatically reduced, and quite possibly eliminated with such a system.

MasterCard's requirements for catalog access, creation and management were:

- Users access product information without going to the Internet.
- Users find products without knowing the supplier.
- The system accommodates all suppliers, regardless of size or technical sophistication.
- The system does not require suppliers to buy new technology in order to do business with the company.
- The system allows for the adoption of new catalog maintenance methodologies.
- Easy catalog maintenance.

In order to satisfy these diverse requirements, Clarus E-Procurement supports a comprehensive, hybrid approach to catalog access, creation and management.

To allow access to product information without going to the Internet, the cross-supplier catalog is situated on MasterCard's trusted intranet, protected by a firewall. This permits quick, secure access to the information required for decision making.

Finding the right product is very easy. Users can search by partial or complete part numbers, keywords, suppliers or manufacturers. Additionally, any of these keys can be combined to refine the search.

MasterCard's suppliers do not wish to invest in non-standard solutions. Tier One suppliers want to leverage existing investments in their Web sites, while Tier Two and Tier Three suppliers want to support existing solutions without investing in proprietary technology.

For these reasons, Clarus E-Procurement supports suppliers of varying levels of technical sophistication and offers multiple methods of cross-supplier catalog population:

- *Manual*

  Basic data entry, directly into the application.

- *Import*

  Cross-supplier catalog content in various electronic forms—including CD-ROM, Microsoft Excel, and standard ASCII file formats such as Catalog Interchange Format (CIF).

- *Dynamic*

  In early 1999, Clarus E-Procurement is integrating CenterStage from OnDisplay for dynamic catalog content management, thus eliminating any manual or scheduled batch update processes.

A single administrator maintains the catalog through the Clarus Catalog Manager. The application's graphical user

interface (GUI), built with Microsoft Visual Basic®, allows for single item entry and update, as well as catalog updates, with the ability to determine what type of update is performed.

This combination of technologies eases the burden of catalog management on MasterCard while eliminating the need for suppliers to invest in new technology in order to do business with the company—although they can realize significant benefits by doing so.

Configuration, Maintenance and Use

MasterCard required that the electronic procurement system be maintainable by one, functionally-oriented individual, rather than many IT professionals. They did not want to focus precious IT resources on an application that was not mission critical—even though the ROI more than justified its implementation.

Clarus E-Procurement includes a full suite of tools that address this requirement. Together, the Clarus Enterprise Manager (CEM) and the Clarus Catalog Manager (CCM) permit non-IT professionals to effectively administer, configure, monitor and maintain the system.

Through its GUI, created with Microsoft Visual Basic, an administrator uses CEM to:

- Configure the user interface
- Create and maintain user profiles
- Create and maintain supplier profiles
- Populate and maintain enterprise information, such as accounting codes, etc.
- Add, change and delete business rules and approval workflow

CCM creates and maintains cross-supplier catalogs. It provides the functionality to enhance catalog data, so that casual end-users can quickly and easily find the correct products or services. CCM includes import utilities, which facilitate catalog updates from a simple modification, such as a price change on a single product from one supplier, to a full catalog refresh. A rollback feature allows administrators to undo improperly applied changes to the catalog.

While the version of Clarus E-Procurement used during the pilot program had an efficient and functionally rich user interface, end-user feedback led to an improved second-generation product now being implemented at MasterCard. This includes a new user interface that is at once more aesthetically pleasing and more intuitive. Now, instructors spend more time teaching users about process change and less on the product itself. Training sessions last an hour, including a question and answer period, for 20 to 25 people.

In addition to these usability goals, MasterCard wanted to achieve a specific objective: The ability for a casual end-user to create and submit a purchase requisition within 30 seconds. To accomplish this goal, the product offers an event-driven procurement model.

In this model, administrators and end-users can create requisition templates that include all products and services that are appropriate for a given event—new employee hire, etc. Within Clarus E-Procurement, these requisition templates are referred to as hot lists. These can be either public or private. As the names imply, public hot lists are accessible by all users, while private hot lists are just that, accessible by only the creator.

When end-users create requisitions, they use an electronic shopping basket. Another type of requisition template, a stored basket, complete with item quantity, account code distributions and other information, can also be created here. This is done most frequently for routine orders—a bimonthly office supply order, for example. Like hot lists, stored baskets can be either public or private.

Whether hot lists or stored baskets are employed, the majority of MRO purchases are driven by an event, and the use of these templates allows the system to meet MasterCard's requirement for lightning fast requisition creation and submission.

Operations

Reporting

The E-Procurement client includes common user reports, which requisitioners use to report and track their own expenditures. However, MasterCard's Purchasing Department generates additional, more complex reports using Microsoft Access and an ODBC connection to the E-Procurement Microsoft SQL Server database. Using various criteria, such as number and dollar total of transactions, by supplier, per month, MasterCard's Purchasing Department can report and analyze:

- The average dollar expenditure per transaction per supplier and the average overall dollar expenditure per transaction.

- The average number of line items per transaction per supplier and again an overall average number of line items.
- The order processing time from submission of request to approval and transmission to the supplier, categorized by commodity.

Troubleshooting

Since most common errors relate to the configuration of Clarus E-Procurement, an E-Procurement administrator resolves 99% of troubleshooting issues. Occasionally additional help is required from IT staff or Clarus Customer Service.

If the issue is related to:

- *Orders*

  Users call the appropriate MasterCard buyer for the commodity/supplier.

- *Catalog*

  For problems with catalog data, the E-Procurement administrator contacts the appropriate supplier(s). For all other issues, they call Clarus Customer Service.

- *Server*

  The E-Procurement administrator contacts MasterCard's IT Department contact for problems with Windows NT Server, Internet Information Server, SQL Server, E-mail, etc. They call Clarus Customer Service application-level issues.

The E-Procurement server generates messages during the workflow process. Key messages indicating status of the workflow are displayed on the server monitor screen. Error messages are also sent via e-mail to internal designees such as the E-Procurement Administrator and IT Administrator. Optionally, these contacts can also be paged.

Maintenance and Backups

Automated and routine virus scanning and database backups comprise the bulk of maintenance for the E-Procurement server. Occasional file purges help maintain adequate disk space. Also, the IT administrator routinely checks the server with remote monitoring and administration utilities.

Manual backups complement the automated routines. These are performed before any significant operation is executed, such as importing major catalogs and updating data sets with scripts.

Staging

A test server stages the new product upgrades. Future plans include using the test server to prototype and stage changes on same-version product, including enterprise configuration changes, users, suppliers, and routing rules.

Back-end Integration

Like most Fortune 2000 companies, MasterCard has multiple back-end systems from several vendors. In addition to various internally-developed legacy systems, MasterCard is currently in the process of implementing a commercial ERP system.

MasterCard requires that the electronic procurement solution have the ability to seamlessly integrate with its heterogeneous back-end environment. Moreover, they require that the system provide a level of abstraction and isolation from the back-end systems. That way, the electronic procurement system can continue to function without modification as back-office applications are upgraded.

Clarus E-Procurement satisfies this requirement through a message-based integration strategy that leverages Microsoft Transaction Server (MTS) and Microsoft Message Queue Server (MSMQ). This loosely coupled approach enables MasterCard to affect integration with multiple, heterogeneous back-office systems in near real-time in an extremely cost-effective manner.

Solution Implementation

MasterCard's original procurement environment was a combination of Microsoft Access databases and Lotus Notes. The system provided electronic forms-type messages for certain types of purchases, coupled with some basic workflow routing. MasterCard required an integrated solution that included:

- Catalog-based product selection based on easy implementation
- Broad support for suppliers, regardless of their level of technical sophistication

- Automated user profile creation
- Event-driven purchasing (new employee hire, etc.)
- End-user requisition tracking
- Highly flexible, yet manageable, business rules administration
- Multiple site deployment with organizational separation
- Standards-based, yet forward-looking, electronic communications with suppliers
- Robust management reporting
- Integration with existing and future back-end applications

And, of course, it all had to work within their existing technical environment.

To accomplish this, the solution was developed using a Microsoft-centric architecture. This was done for several reasons, including Microsoft's commerce strategy and the development of an industry-leading vision of supplier integration. Microsoft Site Server Commerce Edition with the Commerce Interchange Pipeline (CIP) is key to delivering on this vision today and in the future.

MasterCard sought a partner that would allow it to implement a solution immediately while keeping abreast of industry and technology trends. Implementing support for CIP made possible an integrated and secure communications session between MasterCard and its suppliers. As more of MasterCard's suppliers take advantage of CIP, they will be able to create near real-time communications sessions over which to transact business—with both parties realizing many benefits in the process.

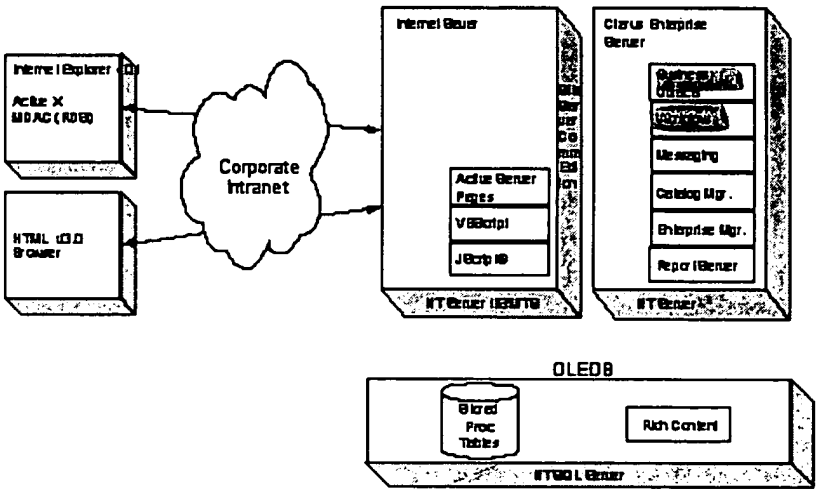
A key driver of this architecture is the need to satisfy the functional requirements and technology issues—not only for the internal procurement process, but for supplier and back-office integration as well. Since the requirements and technologies differ significantly between these three domains, each is addressed in turn.

Internal Procurement

Logical Architecture

Figure 1 illustrates the logical architecture of Clarus E-Procurement as used at MasterCard.

Figure 1: Clarus E-Procurement Logical Architecture At MasterCard



All the MasterCard transaction, profile and catalog index data are maintained in one Microsoft SQL Server database. The schema contains tables for entities such as user, supplier, catalog item, routing rule, order, and report. Purchase order attachments and rich content information (pictures, etc.) are stored in a specified file structure outside the SQL Server database.

MasterCard administrators use two client applications to configure the system:

- **Clarus Enterprise Manager** performs such tasks as creating users and their roles, setting up approval routing rules and setting up supplier information.

- **Clarus Catalog Manager** creates and maintains catalogs.

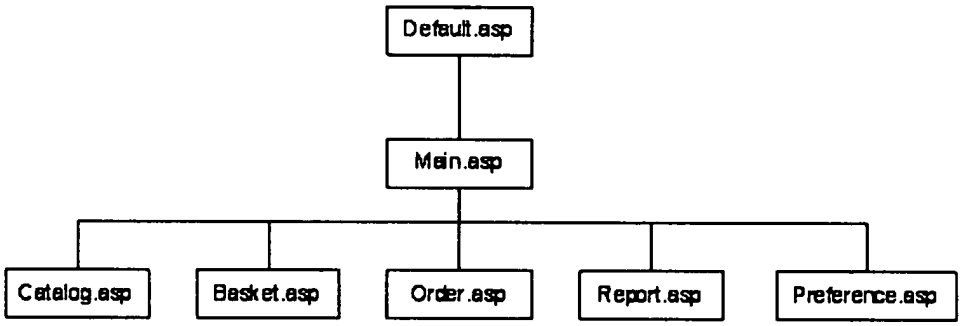
All end-users interact with the procurement system through Web browsers. Virtually all MasterCard users are on Microsoft Internet Explorer 4.0x. As a result, these users view ActiveX® components served through Active Server Pages. Users with older, or non-Microsoft, browsers view the HTML 3.0 version of the application. Since Clarus E-Procurement takes advantage of the performance and component characteristics of ActiveX, users with Microsoft Internet Explorer 4.0x enjoy a much richer experience.

MasterCard chose to access Clarus E-Procurement via its existing Lotus Notes desktop. Once requested, Clarus E-Procurement:

- Displays the user interface
- Controls navigation through the procurement process
- Manages the interactions between both the client- and server-side business objects and the database

The organization of ASP's in the Clarus E-Procurement Application Server mirrors the end-user interface. Figure 2 offers a high-level view of this organization.

Figure 2: High-Level Active Server Pages Organization



All users enter Clarus E-Procurement through Default.asp. This page prompts for user name and password, then starts the session. After querying the database for user information, it sets session variables, and determines the user's browser type and version. Main.asp is opened as appropriate to the browser type and version (ActiveX or HTML).

Main.asp creates a frameset at the highest level of the browser interface, and houses scripts used throughout the application. The user can set the look of the frames.

Five tabs are always displayed: Catalog, Shopping Basket, Orders, Reports and Preferences. Each represents an ASP or group of pages. When a tab is selected, the functionality available on any given page depends on the user's authorized role.

When the user submits a requisition, Clarus E-Procurement Workflow routes it to the appropriate approver(s) via e-mail. Once all approvals are received, purchase order(s) are transmitted. Available transmission methods include hardcopy, e-mail, fax, output of an EDI-friendly text file, or Commerce Interchange Pipeline (CIP). Processing intervals for Workfloware configured through Clarus Enterprise Manager.

Physical Architecture

Components for the application, workflow and database may be deployed on the same or physically separate servers. This provides for scalability, as well as for logical partitioning based on the enterprise's business requirements. MasterCard currently runs all three components on the same physical server.

Security

Clarus E-Procurement uses the Microsoft Windows NT and SQL Server security models for user access. Users must provide an ID and password to gain access to the network and the application. IIS and SQL Server manage data security.

E-Procurement user profiles indicate functional, access-level authorization. Users may be authorized as a:

- **Requisitioner** able to create and submit requisitions and receive against their own orders.
- **Approver** able to approve orders based on approval routing rules.
- **Receiver** able to receive against all orders.

- **Buyer** able to manage catalogs

Load Simulation

MasterCard requires that Clarus E-Procurement support as many as 3,000 users with a 3% to 5% concurrent usage load. About 25% of the organization currently uses the system, with concurrent usage at about 60%.

In February 1998 Clarus performed load simulations using Rational LoadTest 6.1. Testing variables were manipulated to ensure a "lowest common denominator" environment. Specifically:

- The Workflow server was turned off during the simulation, replicating a user option to run the Workflow batch process during non-peak hours.
- The testing application and database servers resided on the same physical Windows NT based-server—a modest Pentium 166 with 192 MB of RAM.
- The testing network was a 100 Mbps Ethernet, and all workstations were equipped with 100 Mbps NIC's.

The test consisted of one "GUI user", a real user for whom system responsiveness was measured, and a number of "virtual users." The simulation software only allowed the GUI user to interact with ActiveX controls. The additional virtual users provided an easy means to simulate load with minimal hardware.

The GUI user repetitively performed the following tasks:

- Log in
- Search on a part number
- Add the part to the shopping basket
- Submit the requisition
- Change a preference
- Log out

The GUI user averaged 45 seconds to complete each cycle in no-load environment.

Virtual users also repetitively performed a sequence of tasks. However, because virtual users could not interact with ActiveX controls, they performed the following tasks:

- Log in
- Read a record from the database
- Log out

The GUI user averaged 120 seconds to complete each cycle when the system was stressed under load.

In general, these tests verified acceptable performance at anticipated load levels.

Technologies

To build the optimal solution that affords MasterCard the opportunity to obtain a rapid return on investment, while maintaining upgrade flexibility, the application uses the following technologies:

Dynamic Web pages, employing ASP, HTML, JavaScript, and VBScript

- Client-side ActiveX controls
- Active Data Objects (ADO)
- Remote Data Services 1.5 (RDS)
- Windows NT Server 4.0
- Internet Information Server 4.0 (IIS)
- SQL Server 6.5
- Site Server 3.0 Commerce Edition

Third-Party Components

MasterCard requires end-users, as well as, management reporting. Clarus bundles Seagate Crystal Reports 6.0 with Clarus E-Procurement and includes numerous standard report templates for this purpose. With this technology, reports transmit to the browser, where the Crystal Reports viewer component allows users to search and sort the results.

System Configuration

The following tables list the system configuration for Clarus E-Procurement 3.0 as installed at MasterCard International.

Application Server	
Operating System	Microsoft Windows NT Server 4.0 (SP3)
Internet Server	Microsoft Internet Information Server 4.0 with Microsoft FrontPage® Extensions
	Microsoft Remote Data Services 1.5[1]
Reporting	Seagate Crystal Reports 6.0 (runtime version)[1]
Network Protocol	TCP/IP
E-Mail Client	Lotus Notes 4.6
Hardware	HP Kayak 3000 Dual Pentium II 300 MHz Processor 128 MB RAM 4 GB Hard Drive
Database	Microsoft SQL Server 6.5 (SP4)
Database Connectivity	Microsoft ODBC Administrator 3.5[1]

End-User Workstation	
Operating System	Microsoft Windows NT Workstation 4.0
Web Browser	Microsoft Internet Explorer 4.0x
Network Protocol	TCP/IP
E-Mail Client	Lotus Notes 4.6
Hardware	Pentium 133 MHz or better processor 32 MB RAM 800 x 600, 256-Color SVGA Video Adapter and Display

Administrator Workstation	
Operating System	Windows® 95/98 and Windows NT
Reporting	Crystal Reports 6.0
Database Connectivity	Microsoft ODBC Administrator 3.5
Network Protocol	TCP/IP
Hardware	Pentium 133 MHz or better processor

	64 MB RAM 800 x 600, 256-Color SVGA Video Adapter and Display

System Administrator Responsibilities	
Windows NT Server Administrator	Prepare and Install Clarus E-Procurement server software
Web Master	Web configuration and maintenance
Database Administrator	Performance Tuning SQL Server Monitoring Backups

1 Denotes third party license provided with Clarus E-Procurement.

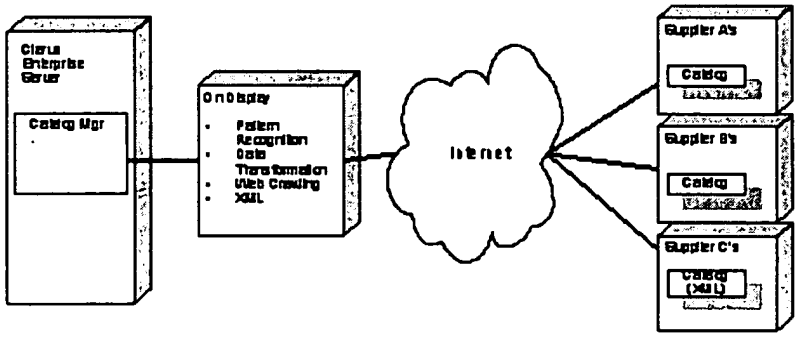
Supplier Integration

For MasterCard, and most other Fortune 2000 companies, supplier integration is a key component of an electronic procurement solution. Clarus addresses supplier integration through both catalog and transactional data. In addition, MasterCard requires that Clarus E-Procurement be able to work with any supplier to minimize the cost of business transactions between the buying organization and its suppliers. Since the number and variety of potential MasterCard suppliers is vast, E-Procurement must offer flexible, low-cost solutions for online transactions and catalog management.

Logical Architecture

The system components associated with delivering supplier integration to MasterCard are illustrated in Figure 3.

Figure 3: Supplier Integration Components



Workflow

Clarus E-Procurement Workflow is responsible for transmitting purchase orders to suppliers. The available transmission methods are hardcopy, e-mail, fax, EDI, and Microsoft Commerce Interchange Pipeline (CIP). Of these, only EDI and CIP offer application-to-application transaction automation.

Historically, EDI provided the only option for application-to-application integration of business transactions. However, EDI is expensive and labor intensive. To use EDI, both MasterCard and its suppliers would be required to implement translators and establish relationships with Value Added Network providers. Although MasterCard and some of its suppliers are moving in this direction, it is not the most cost-effective or efficient solution for either party given today's Internet technologies. Microsoft Site Server Commerce Edition with CIP provides the first commerce framework that shows significant promise to reduce the reliance on EDI for automated transactions over the Internet.

Catalog Manager

Clarus Catalog Manager (CCM) is used by MasterCard's administrator to create and update catalogs. It uses a fixed-

format text file to update the catalog table in the E-Procurement database. To convert the different file formats provided by suppliers into a fixed-format text file usable by CCM, Clarus E-Procurement supports CenterStage from OnDisplay.

Online Transactions

As mentioned earlier, MasterCard supports multiple purchase order delivery mechanisms for its suppliers. Hardcopy, fax, e-mail, EDI and CIP are all utilized. The TransmitVia component in the application directs transactions to the correct delivery format as outlined below:

- **TransmitViaEDI** directs a formatted file with pre-defined EDI data elements to an EDI translator for transmission to suppliers.
- **TransmitViaE-mail** directs a Crystal Reports-formatted attachment to the e-mail server at MasterCard (Lotus Notes) with the address definition required for delivery to the supplier's order entry desk.
- **TransmitViaFax** directs a Crystal Reports-formatted attachment to the fax gateway at MasterCard with the address definition required for delivery to the supplier's order entry desk.
- **TransmitViaPrint** routes a Crystal Reports-formatted document to a printer for delivery via postal mail.
- **TransmitViaCIP** transmits structured data to a supplier through Microsoft Commerce Interchange Pipeline (CIP). CIP allows applications to freely exchange information without respect to transport protocols and data formats.

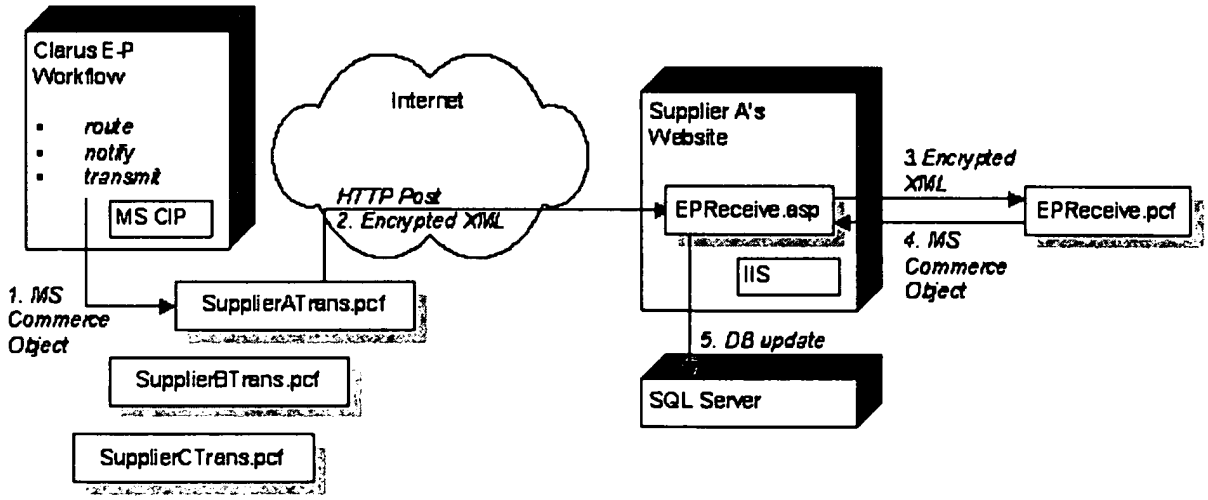
  CIP manages the transport and data format details and is easily configured using Microsoft's graphical Pipeline Editor.

  TransmitViaCIP allows administrators to define the data source, fields, field names, and field hierarchical organization. This minimizes transaction costs, because a different purchase order document structure may be defined for each supplier for repeated use.

Microsoft Commerce Interchange Pipeline

Figure 4 illustrates one scenario in which applications at separate trading partners exchange data via CIP. In this scenario, Clarus E-Procurement Workflow transmits a purchase order to Supplier A.

Figure 4. Data Exchange Via CIP



Step 1: Workflow's transmit step creates a Microsoft Commerce Dictionary object, and populates it with purchase order data. The transmit step then invokes the transmit pipeline configuration file for Supplier A (SupplierATrans.pcf) and passes it the populated dictionary object

Step 2: The transmit pipeline configuration file knows what data format, encryption method, and transport protocol to use. It formats the data in XML, encrypts it, and performs an HTTP Post to the EPReceive.asp at Supplier A's Web site.

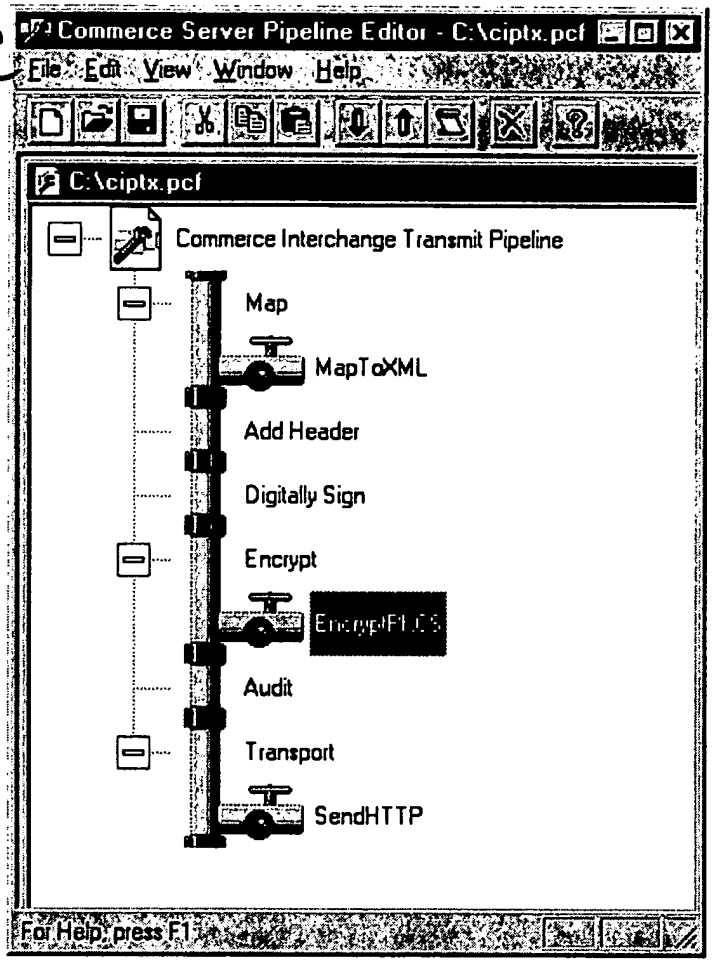
Step 3: The receiving ASP creates a Microsoft Commerce Dictionary object invokes the receive pipeline configuration file (EPReceive.pcf), passing it the dictionary object and the incoming data.

Step 4: The receive pipeline configuration file knows what encryption method and data formats were applied to the incoming data. The receive pipeline receives the incoming data, decrypts it, parses the XML, populates the dictionary object with the data and passes the dictionary object back to the ASP script.

Step 5: The ASP script navigates and processes the populated dictionary object as appropriate. In Figure 4, the ASP script updates the supplier's database.

Figure 5 shows the creation of a transmit pipeline configuration file in Microsoft's Pipeline Editor. The pipeline consists of a sequence of stages (Map, Add Header, Digitally Sign, Encrypt, Audit and Transport), for which a choice of components are provided. A simple right-mouse click adds a component at any stage.

Figure 5 CIP Editor



Fundamentally, CIP serves to reduce the size and complexity of the application's architecture by abstracting data transport processing. Note how CIP simplifies the requirements of dealing with multiple suppliers. In Figure 4, if the purchase order is sent to Supplier B instead of Supplier A, Workflow's transmit step merely invokes SupplierBTrans.pcf instead of SupplierATrans.pcf. The Workflow application knows nothing about the data format, encryption and transport methods employed in either case.

Clarus TransmitViaCIP

Clarus TransmitViaCIP allows the administrator to define document structure, data source, included fields, field names, and hierarchical field organization. In addition, Clarus E-Procurement provides for file attachments to be transmitted along with the document.

The administrator defines a new document structure, or "schema", by making entries in database tables. A schema consists of a hierarchical organization of "document objects", each of which represents a logically related group of data items. For example, document objects in a purchase order schema typically include:

- Order header
- Dates
- Requisitioner
- Shipping

- Billing
- Line items
- Attachments.

Data required for each document object include:

- Unique ID
- Name
- Type
- Name of stored procedure that will be executed to obtain the data for populating the object
- Object parent ID of the object's parent (to establish the hierarchical structure).

In addition, "friendly" field names can be specified so that the name transmitted for any given field can differ object schemas.

Three data types are used for document objects:

- **Dictionaries** are Microsoft-provided "Commerce objects." They are containers that hold attribute/value pairs, where values are any type of object.
- **SimpleLists** are Microsoft-provided "Commerce objects." They are containers that hold unnamed objects that can be enumerated in code or script.
- **Attachments** are Clarus-provided objects. They are used specifically for representing file attachments.

Because a Dictionary can contain Dictionaries and SimpleLists, and a SimpleList can contain Dictionaries and SimpleLists, it is straightforward to represent arbitrary hierarchical data using them.

The interface to TransmitViaCIP is:

```
Public Function Process(vDataSource As Variant, _
vDataKey As Variant, _
vCIPSchemaID As Variant, _
vPipelineConfig As Variant) As Boolean
```

The Boolean return value indicates success or failure of the transmit operation. The parameter meanings are as follows:

- **vDataSource** specifies which database contains the document data.
- **vDataKey** specifies the ID that uniquely identifies the document to be transmitted. For example, the purchase order number identifies a purchase order.
- **vCIPSchemaID** specifies the ID that uniquely identifies the administrator-defined schema to use. For example, this might specify the purchase order schema for SupplierA.
- **vPipelineConfig** specifies which transmit pipeline configuration file to use. For example, this might be SupplierATrans.pcf.

Clarus Workflow Transmit Step Example:

The administrator previously used Clarus Enterprise Manager to specify that purchase orders for Supplier A should be transmitted via CIP, and identified the schema ID as well as the transmit pipeline configuration file.

When the transmit step encounters a purchase order for Supplier A, TransmitViaCIP is invoked and the data source, purchase order number, schema ID, and the name of the transmit pipeline configuration file are passed to it. TransmitViaCIP then performs the following tasks:

1. Schema is read (using vCIPSchemaID) from the database (vDataSource)
2. Document structure is built using Dictionary, SimpleList, and Clarus Attachment objects as specified
3. Stored procedures specified in the schema are executed to obtain the purchase order data (using vDataKey) from the database (vDataSource)

4. File attachments are read
5. Document structure is populated with the purchase order data and any file attachments
6. Root document structure object is passed to the transmit pipeline (vPipelineConfig)
7. Root document structure object is transmitted to the supplier
8. Return result is passed to the transmit step of Clarus Workflow

Supplier Technical Requirements

Clarus E-Procurement imposes no requirements on suppliers. However, in order to realize the benefits available by using CIP, suppliers must implement Microsoft Site Server Commerce Edition. Additionally, CIP and the TransmitViaCIP object included free of charge with Clarus E-Procurement must be configured and enabled.

The default installation of Clarus TransmitViaCIP assumes that HTTP Post is employed as the transmit method in the pipeline configuration file. It further assumes that the supplier is running Microsoft Internet Information Server so that received documents can be navigated and processed with ASP scripts that are provided in near-complete form by Clarus.

In this scenario the supplier requires the following components:

- Microsoft Internet Information Server
- Microsoft Site Server Commerce Edition
- The Clarus ReceiveViaCIP component to decode file attachments
- The EPReceive.pcf receive pipeline configuration file. This must be compatible with the **transmit** pipeline configuration used by the buying organization.
- The EPReceive.ASP script. This ASP, which is included with Clarus E-Procurement, contains the script to call the receive pipeline, and to navigate and process the data structure. This must be compatible to the schema defined at the buying organization.

Catalog Management

Another critical consideration for MasterCard is the ability to support a diverse set of Catalog Management requirements. MasterCard successfully automated its content management process by working with key suppliers to support both direct data entry by administrators and batch file updates. The Clarus Catalog Manager provides an easy-to-use interface to support the content management requirements for an unlimited number of suppliers.

OnDisplay CenterStage ECat

The next version of Clarus E-Procurement supports CenterStage ECat from OnDisplay for automated, proactive catalog content management.

CenterStage ECat was chosen for its ability to easily create and maintain buy-side, cross-supplier catalogs. Its features include:

- An easy-to-use, point-and-click interface
- The ability to access and transform data from a wide variety of sources including flat ASCII files, application files (i.e. Excel, Word, etc.) and reports, native database formats (i.e. Access, DB/2, dBase, Visual FoxPro®, SQL Server, etc.), Web pages (i.e. HTML/XML), even e-mail messages
- Pattern recognition to identify elements within a data source

Dynamic Catalog Content Management Example:

To initially populate the Clarus E-Procurement cross-supplier catalog, the administrator uses CenterStage ECat to perform an "initializing navigation" of electronic catalogs provided by suppliers. During this process, procurement-relevant fields are identified for a single item in each catalog. This establishes the patterns of data. CenterStage ECat then automatically navigates supplier catalog(s) to initially populate, and subsequently update, Clarus E-Procurement's cross-supplier catalog. This process is the same for all supported data sources.

Since Clarus Catalog Manager can run in command-line mode using pre-defined catalog update profiles, MasterCard

can schedule CenterStage ECat and Clarus Catalog Manager to update catalogs automatically.

Back-end Integration

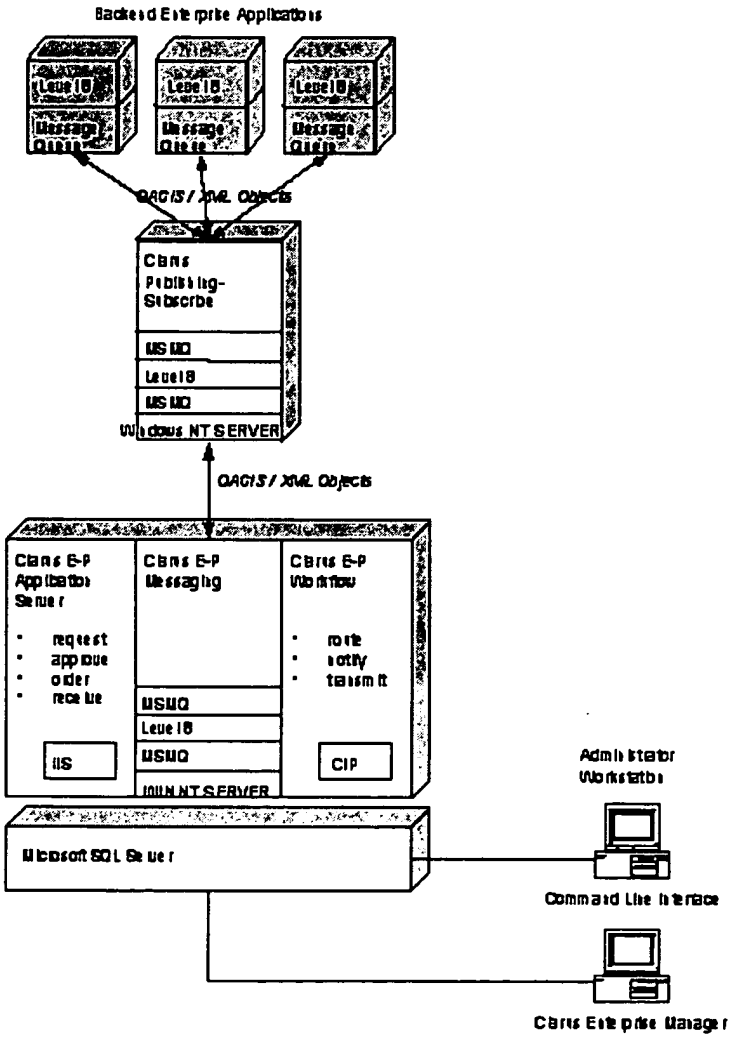
Clarus E-Procurement must integrate with a variety of back-end systems at MasterCard. This requirement, which is common to all Clarus E-Procurement customers, led to three architectural design principles:

- **Loose coupling** to ensure that changes to back-end systems did not effect Clarus E-Procurement.
- **Broad applicability** to support integration with any back-end system, past, present and future.
- **Inter-application workflow** to permit the creation of total business solutions from isolated, line-of-business applications.

Logical Architecture

The Clarus E-Procurement Messaging component that handles both incoming and outgoing messages to affect back-end systems integration is shown in Figure 6.

Figure 6: Back-end Systems Integration Components



The Messaging component creates outgoing messages for any event within Clarus E-Procurement that needs to be communicated outside of the application. Likewise, the Messaging component receives and manages the processing of incoming messages, which are generated by other applications and represent events that need to be communicated to Clarus E-Procurement.

As an example of an outgoing message, Clarus E-Procurement notifies back-end systems whenever Workflow transmits purchase orders to suppliers. This enables, among other things, a purchasing control application to prepare for subsequent invoice matching and payment processing.

As an example of an incoming message, Clarus E-Procurement updates its table of Supplier data in response to messages sent by other applications. This enables an administrator to manage Supplier data in a single back-end system and the data are automatically synchronized in the Clarus E-Procurement database.

The Clarus Publish-and-Subscribe component is a hub through which all messages flow. It enables applications to tap into this flow and retrieve only relevant messages. For example, a purchasing control back-end application "subscribes" to receive information on all purchase orders. Subsequently, the Publish-and-Subscribe component sends a copy of all purchase order messages to the purchasing control application.

Physical and Network Architecture

The Clarus E-Procurement Messaging component is deployable on the same or physically separate servers as the Application, Messaging, Publish-and-Subscribe, Workflow and database servers. All of these components, as well as the administrator applications, must reside on the same intranet. Future implementations of the Publish-and-Subscribe component will allow applications to connect via the Internet, so that the servers can physically reside anywhere in the world.

Message Objects

In order to avoid an explosion of translations between the business objects at each back-end system, it is highly desirable to have a standardized message set for the exchange of business data between applications. To this end, the Clarus E-Procurement Messaging component employs an OAGIS-compliant message set, with the messages formatted as XML.

OAGIS

The Open Applications Group is a nonprofit consortium of enterprise application software developers. The organization was formed to create common standards for the integration of enterprise business applications. OAG members include: J. D. Edwards, Oracle, PeopleSoft and SAP.

The Open Applications Group Integration Specification (OAGIS) provides a standard set of messages for the exchange of structured business data.

XML

Extensible Markup Language (XML) is a structured data format. XML provides a simple and powerful way of representing arbitrary hierarchical data. The emergence of a standard API for processing XML documents—the Document Object Model, or DOM—and the availability of generic tools for processing XML-formatted documents both convey significant architectural advantages to using XML.

Use of OAGIS and XML together confers a powerful advantage beyond minimizing translations between applications. Use of XML means that message data is highly accessible, and use of OAGIS means that both the business process context of messages and the meaning of data within them are known. As a result, the Publish-and-Subscribe component can discriminate between messages flowing through it with great sophistication. It will also be able to relate messages to business processes occurring within the individual interoperating applications. This is where future inter-application workflow will be implemented.

Technologies

The movement of messages into and out of the Messaging component is through accomplished by using queuing technology—specifically Microsoft Message Queue Server (MSMQ). Beyond MSMQ, Clarus Fusion provides a powerful range of capabilities for enterprise application integration in messaging environments, with sophisticated XML capabilities.

Message queuing enables loose coupling of applications while guaranteeing the delivery of messages. Rather than procedure calls, with APIs that differ vastly between applications, communication is achieved by through the exchange of messages using agreed upon formats and transports.

The queuing system guarantees messages delivery, even if receiving applications are unavailable at the time messages are generated and sent. Sending applications do not have to wait for responses from receiving applications before continuing processing.

Clarus Fusion handles the interface between Clarus E-Procurement's internal business objects and the message queue. For incoming messages, Fusion monitors the queue, picks up arriving OAGIS/XML messages and parses the XML. Based on the content of the parsed XML, a script determines the type of OAGIS message and invokes a corresponding workflow. The workflow, in turn, transforms the OAGIS data into the form required by Clarus E-Procurement's business objects and passes it to the business objects. Since both Clarus Fusion and E-Procurement objects are designed to work with Microsoft Transaction Services (MTS), processing of entire message are handled as

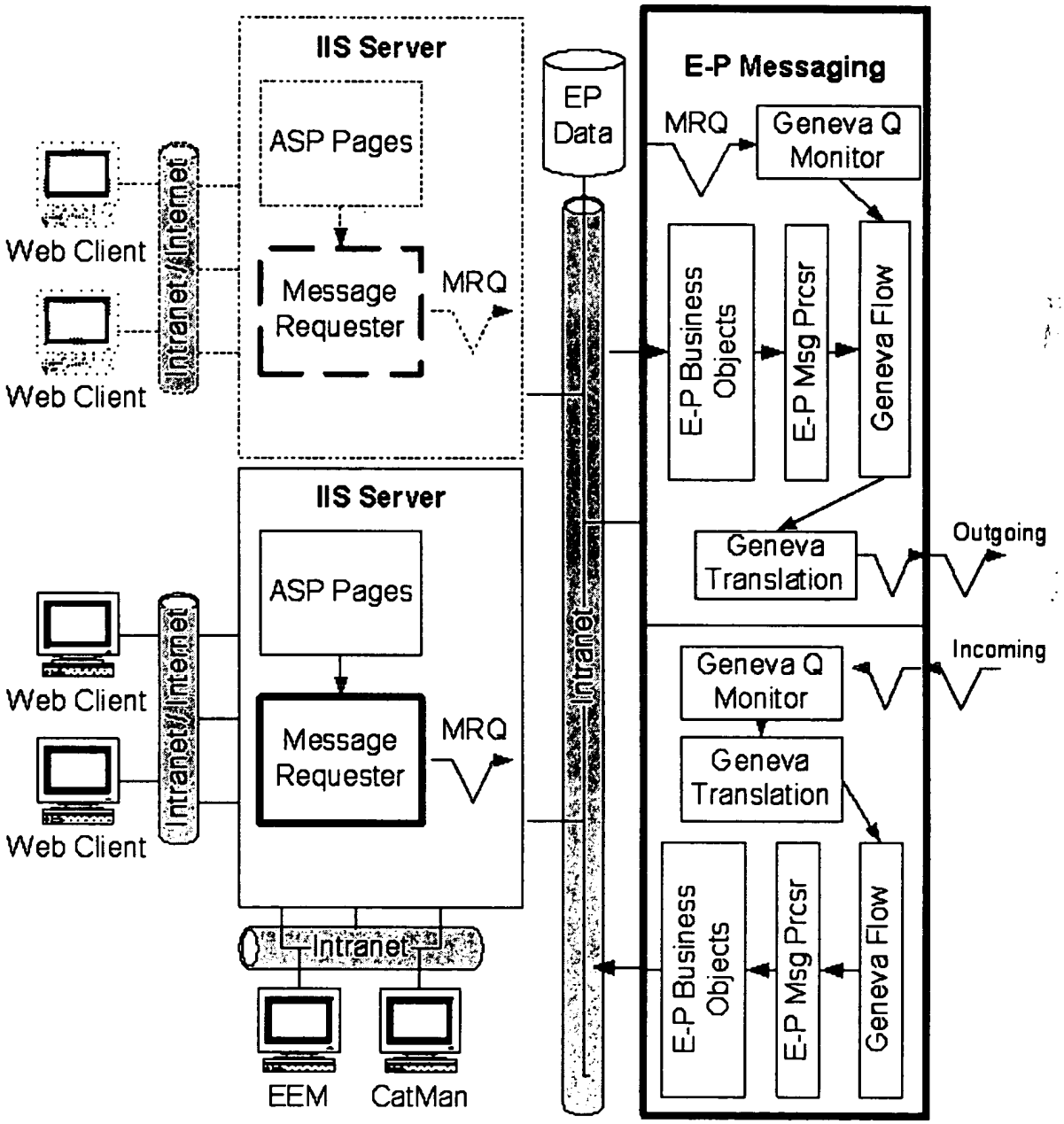
a single transaction. The process is similar for outgoing messages.

Because Clarus Fusion provides such a broad range of capabilities, the internal E-Procurement business objects remain optimized for procurement functionality. The E-Procurement business objects do not have to account for details of the external message set (OAGIS) and formats (XML). Instead, all such adaptation is handled by script and configuration files within Fusion.

Clarus E-Procurement Messaging Component

Figure 7 shows the logical architecture of the Clarus E-Procurement Messaging component. A single representative configuration, relative to other components is shown. Other configurations are possible.

Figure 7: Messaging Component



Interface to the Clarus E-Procurement Messaging Component

The Clarus E-Procurement Messaging component interacts with the Publish-and-Subscribe component through the "Incoming" and "Outgoing" queues. These queues are monitored by the Publish-and-Subscribe component and the messages traversing them are OAGIS/XML messages.

Any component of Clarus E-Procurement (i.e. an ASP, Clarus Enterprise Manager, Clarus Catalog Manger, Workflow, etc.) can initiate the creation of an outgoing message by placing a "request" in the Message Requester Queue (MRQ). The request contains the data necessary to initiate the creation of the correct message.

The format of the request is simple XML:

```
<REQUEST DocType="..." DocKey="..." DataSource="..." ... />
```

Where DocType, DocKey, and DataSource specify the type of document to create, the relevant key value and which database to use, respectively.

In addition, messages are encrypted before they are sent.

Internal Structure of the E-Procurement Messaging Component

The Messaging component employs Clarus Fusion to monitor queues, transfer messages, manage the sequence of processing steps, assemble and handle hierarchical data internally, and translate between internal data formats and OAGIS/XML.

Many OAGIS documents can be handled as hierarchical structures that are built up from simpler business objects, and the message processor orchestrates the operation of individual business objects as necessary to create or process an entire OAGIS document.

The Message Processor may generate a message requests via the Message Requester. This could be the result of business rules associated with normal validation and processing of inbound messages. That is, one business event may trigger another.

Lessons Learned

Many technical and operational lessons were learned throughout the process of implementing Clarus E-Procurement at MasterCard. The technical lessons are rapidly being addressed in upcoming versions of the product to which MasterCard contributes significant input.

- **Adopt leading edge, not bleeding edge technology**

  Clarus adopted a Microsoft-centric technical approach early on, while its competitors focused on Java. Microsoft technologies have enabled Clarus E-Procurement to provide a rapid return on investment to MasterCard, and provide value to both MasterCard and its suppliers through the flexibility allowed by the Microsoft platform and support of CIP.

- **Promote internally**

  It is essential to create a positive aura around any new application, and MasterCard did an outstanding job of making Clarus E-Procurement available for potential end-users to both see and hear about its value to individuals and the company.

- **Re-engineer after implementation**

  MasterCard implemented Clarus E-Procurement as a completely stand-alone system. This has given MasterCard time to gather data about user habits and processes prior to any business process re-engineering efforts, such as the ERP system implementation scheduled for later in 1999.

- **Include suppliers**

  MasterCard and Clarus have both been sensitive to the nature of supplier relationships and have found that approaches to catalog management and supplier integration vary greatly from vendor to vendor. Clarus' approach to catalog management has been a proven success with both MasterCard and its key suppliers.

- **Deploy in phases**

  Although Clarus E-Procurement can be rolled out to a large number of users at once, deployments should be planned in phases within an appropriate time frame. Training is required for end-users, but even more importantly, individuals need time to adapt to the change in process. This is an application that will change the way people do business, and therefore will have more of a social—rather than technical—impact on the individual.

Summary

MasterCard's installation of Clarus E-Procurement was the first implementation of a commercially available Web-based procurement application in the market. The Microsoft infrastructure was critical to the success of the implementation in terms of reliability, scalability and cost of implementation.

Since the initial implementation in June 1997, the system has never failed and has experienced zero downtime—even

through three product upgrades. Clarus E-Procurement requires less than half of one person's time to maintain and administer, and requires the lowest total cost of ownership for any competing solution in the customer segment.

Finally, the availability of Microsoft Site Server Commerce Edition and the Commerce Interchange Pipeline provides Clarus the opportunity to create, and MasterCard to implement, the most compelling electronic procurement solution in the industry. Without the Microsoft technology available in the customer segment, this would not be possible.

About Clarus Corporation

Founded in 1991, Atlanta-based Clarus Corporation, (NASDAQ:CLRS), is a leading provider of Web-based applications for managing operational resources together with Financial and Human Resource applications for mid- to large-sized companies. These applications create high lifetime value by delivering sophisticated functionality while substantially reducing the time required for implementation, maintenance and upgrades.

Clarus Corporation

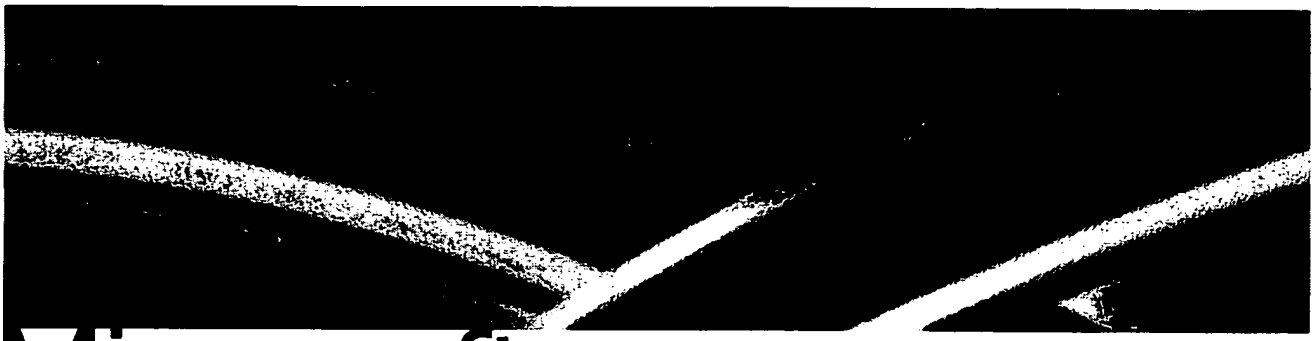
3970 Johns Creek Court

Suwanee, GA 30024

Tel.: 770-291-3900

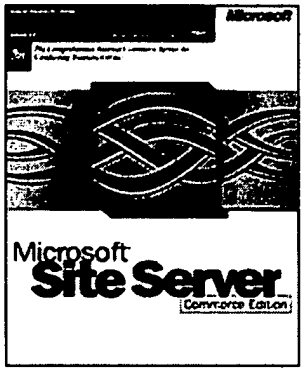
Fax: 770-291-3999

Web site: HTTP://WWW.CLARUSCORP.COM



Microsoft® Site Server Commerce Edition

Comprehensive Internet Commerce Server for Conducting Business Online



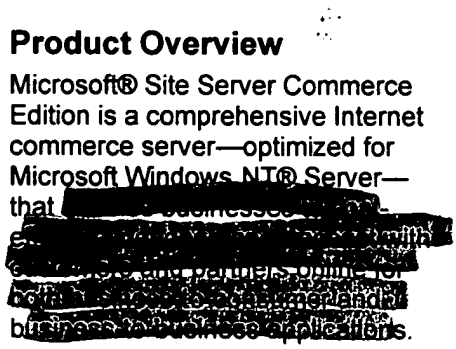
Microsoft® Site Server Commerce Edition is a comprehensive Internet commerce server, optimized for Microsoft Windows NT® Server, that enables businesses to cost-effectively engage customers and transact business online.

Is your business taking full advantage of the World Wide Web?

The World Wide Web profoundly affects the way companies communicate and transact business. The Internet enables companies to reach customers in new and innovative ways. Businesses can build customer loyalty, create an efficient channel for transactions with distributors and suppliers, and deliver their message in an increasingly global market—cost effectively. Companies can capture the attention of customers and partners with targeted online promotions and advertising, transact high volumes of business securely and reliably, and actively manage and analyze their sites to maximize their return on investment.

Product Overview

Microsoft® Site Server Commerce Edition is a comprehensive Internet commerce server—optimized for Microsoft Windows NT® Server—

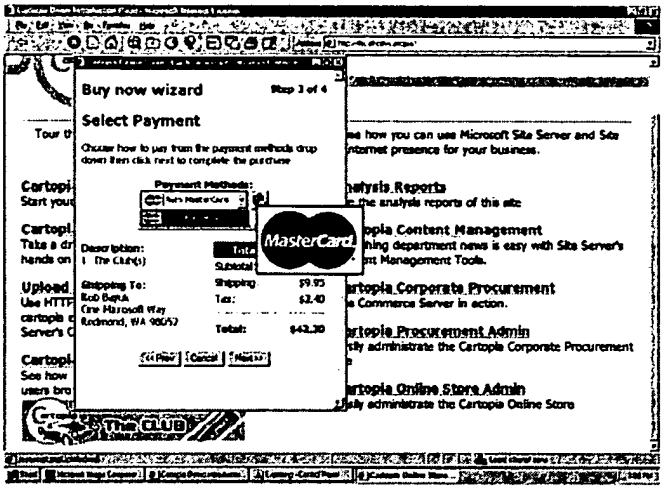


Microsoft Site Server Commerce Edition is a comprehensive Internet commerce server for engaging customers, transacting business, and analyzing eCommerce Web sites. Site Server Commerce Edition helps businesses deploy and manage business-to-consumer, corporate purchasing, and supply chain management applications. By providing a comprehensive set of server components, management tools and sample sites, it significantly reduces development time and costs for these applications. Site Server Commerce Edition enables the sale of goods and services to

customers and business trading partners. You can promote and merchandise products dynamically, run a more efficient online business, derive revenue from online advertising, and understand and improve your business through comprehensive site analysis.

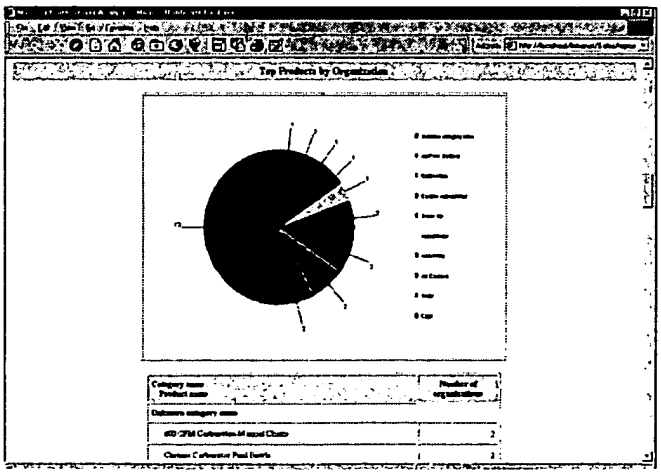
Site Server Commerce Edition provides a comprehensive set of features that can be easily integrated into existing accounting or order management systems. And software from more than 50 independent application vendors makes it possible to extend the platform—with specialized billing, payment, or accounting systems, for example—with less need for custom development.

Whether you are creating a new site or adding commerce capabilities to an existing one, Microsoft Site Server Commerce Edition enables you to:



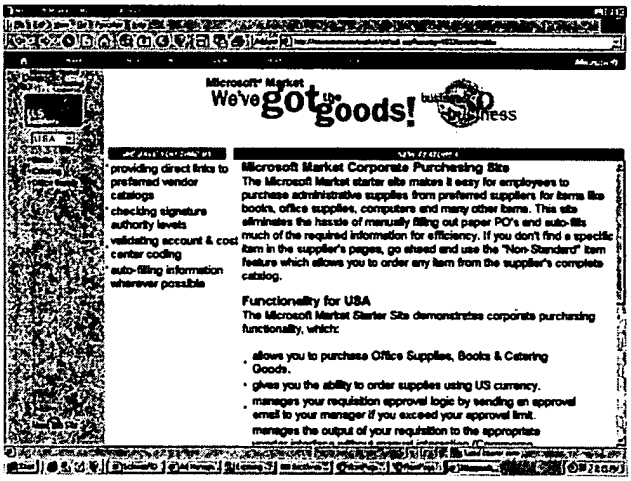
Buy Now and the Microsoft Wallet enables spontaneous purchasing across the Web in just four clicks of a mouse.

Transact business online with secure and scalable [redacted] processes. Streamline your Web business by integrating orders with existing business systems. Route EDI transactions over the Internet directly from your Web application. Provide customers with the highest level of industry-standard security and encryption, ensuring that each transaction meets its proper destination.



Site Server Analysis provides critical insight into web traffic, where people are going and what they doing on your site.

Engage customers and partners
Create and manage dynamic promotions, online advertising campaigns, and personalized Web sites to build enduring relationships with customers and maximize sales. Encourage spontaneous purchases with transaction-enabled content. Establish custom catalogs and pricing for business partners. Manage high volumes of online user accounts for self-service applications.



The Microsoft Market corporate purchasing sample site, provides an example of business-to-business commerce based on Microsoft's own internal corporate purchasing solution.

Analyze usage to understand and improve your business
Actively manage Web content and analyze usage data, regardless of the mix of Web servers utilized. Generate preconfigured analysis reports or mine usage data with custom reports to uncover important insights about site activity. Use any remote Web browser to maintain direct access to product purchase data.

Product Highlights

Microsoft Site Server Commerce Edition includes the following features to help you manage the complete life cycle of your Web site:

Commerce Server	Commerce Server includes a comprehensive set of features, sample sites, and tools that enable you to engage customers and transact business online: – **Site Builder Wizard and Sample Sites** remove the complexity of database schema editing, scripting, and HTML coding. The simple, step-by-step approach of the Site Builder Wizard dramatically reduces site development time. Sample sites provide examples of business-to-consumer and business-to-business commerce applications. – **Dynamic merchandising** provides support for easy, real-time administration of product and price promotions from any remote Web browser through the Promotion Wizard. Intelligent CrossSell uses previous shopper trends to automatically make recommendations. – **Order Processing Pipeline** handles targeted functions—such as product tax, shipping and handling charges, payment authorizations, and inventory checks—according to specific business rules. It can be integrated with existing systems and can also be extended with many products from independent software vendors. – **Commerce Interchange Pipeline** enables applications to exchange information using the Internet or an existing EDI system. Because the Commerce Interchange Pipeline is data format-independent and transport-independent, businesses of all sizes can communicate securely. The Commerce Interchange Pipeline supports native Web formats such as XML and HTTP as well as those from numerous independent EDI software vendors. – **Integration with Microsoft Transaction Server**—a transaction processing system included with the Windows NT Option Pack—allows Site Server to provide site and application developers a business-critical solution that offers significantly higher reliability in business transactions. – **Dynamic catalog generation** creates custom Web catalog pages on the fly using Active Server Pages to directly address the needs, qualifications, and interests of visiting customers. – **Rich object model** manages products, users, and orders. It supports schema and database independence, enabling businesses to integrate their existing business rules and data with their online presence. – **Buy Now,** a powerful online marketing solution, lets you embed product information and order forms in most online contexts—such as online banner ads—for quick, spontaneous purchase by consumers. – **Commerce Host Administrator,** a control center for site administrators and Internet hosting service providers, enables the centralized administration of multiple commerce sites while allowing individual site managers to update their sites remotely – **Built-in Microsoft Wallet** support helps businesses provide customers with the most convenient and secure online payment experience. – **Industry-standard security** creates a secure environment for customers, partners, and site/application administrators with strong, integrated HTTP Authentication and Windows Challenge Response. Site Server Commerce Edition supports real-time credit authorization with secure transaction protocols such as SSL and Secure Electronic Transaction (SET). – **Commerce Server Software Developer's Kit (SDK),** a set of open application programming interfaces (APIs) allows full extensibility across the entire Order Processing and Commerce Interchange Pipelines.

Personalization and Membership	Membership lets you easily manage users and user profiles for high-volume sites. Secure access to any area of the site, supporting subscription or "members only" applications. Personalization enables the delivery of custom content based on the site visitor's personal profile and supports targeted promotions and one-to-one marketing. – **Direct Mailer** is an easy-to-use tool for creating a personalized direct e-mail marketing campaign based on Web visitor profiles and preferences. – **Membership Server** provides the software infrastructure for efficiently managing secure access to Web sites and site content, with the ability to scale to support millions of visitors. Authentication can be based on cookies, Basic and HTML forms, challenge/response, and certificates.
Ad Server	Ad Server manages ad schedules, customers, and campaigns through a centralized, Web-based management tool. Target advertising to site visitors based on interest, time of day or week, and content. In addition to providing a potential source of revenue, ads can be integrated directly into Commerce Server for direct selling or lead generation.
Site Server Analysis	The Site Server Analysis tools let you create custom reports for in-depth analysis of site usage data. Create industry-standard advertising reports to meet site advertiser requirements. Classify and integrate other information with Web site usage data to get a more complete and meaningful profile of your visitors and their behavior. Enterprise management capabilities enable the central administration of complex, multihomed, or distributed server environments. Supports 28 Web server log file formats on Windows NT, UNIX, and Macintosh operating systems, including those from Microsoft, Netscape, Apache, and O'Reilly. – **Commerce Order Manager** gives direct access to real-time sales data on your site. Analyze sales by product or by customer to provide insight into current sales trends or manage customer service. Allow customers to view their order history online.

System Requirements

- Either an Intel Pentium 100 MHz or higher processor (Intel Pentium 166 MHz or PRO recommended) or RISC-based system with an Alpha processor
- 64 MB of RAM (128 MB recommended)*
- 1 GB of available hard disk space (2 GB recommended)*
- CD-ROM drive

Additional software requirements:

- Microsoft Windows NT Server version 4.0 and Windows NT 4.0 Service Pack 3.0 or later
- Windows NT 4.0 Option Pack (included)
- Microsoft SQL Server™ or ODBC 3.0-compliant database required for Commerce Server and Ad Server
- Microsoft FrontPage® Web site creation and management tool (included)
- Microsoft Visual InterDev Web development system (included)

* Actual requirements will vary based on system requirements and recommendations for certain Web-based applications.

If you intend to use Commerce Server to support multiple domains you must acquire a Domain Access License (DAL) for each additional domain. The DAL can only be used with the Microsoft Commercial Internet System 2.0, which includes Microsoft Site Server Commerce Edition 3.0 and additional features for supporting hosted environments. A "domain" shall mean the description of a computer's location on the Internet comprised of a second-level domain name (for example, microsoft) and a top-level domain name (for example, com). For example, http://www.microsoft.com and http://www.microsoft.com/products are part of a single domain, while http://www.microsoft.com and http://www.msn.com are not.

For a complete list of client system requirements and recommendations for certain Web-based applications, visit the Microsoft World Wide Web site at http://www.microsoft.com. For Microsoft Site Server Commerce Edition pricing and licensing information, visit the Microsoft World Wide Web site at http://www.microsoft.com/siteserver/commerce/.

Site Server Commerce Edition 3.0 is a superset of the Microsoft Site Server server product. It includes all of its features for dynamically publishing content, searching content and the delivery of content in multiple formats.

Microsoft Corporation • One Microsoft Way • Redmond, WA 98052-6399 • USA
0498 Part No. 098-80305

The Library
- Case Studies
- Supply-chain Trends
- SCM Technology
- Logistics/Transportation
- Interviews/Opinions
- Facility Location Planning

Newsletters
- QuickREAD
- e-INSIDER
- MarketWatch
- New Business Report

Online Magazines
- Logistics Outsourcing
- Warehouse Management
- Transportation Mgmt.

Directories
- SCM Technology
- Logistics/Transportation
- Facility Location Planning
- Regional Logistics
- Consultants
- Educational Institutions
- Events Calendar

Consulting Services
- SupplyChainXchange
- Market Intelligence
- Jump Start
- Technology Evaluation

SCM Research
- ARC Advisory Group
- Aberdeen Group
- AMR Research

Global Logistics & Supply Chain Strategies [redacted]

Trading Exchanges Have the 'Big Mo,' But Users Should Proceed With Caution

By Paul Korzeniowski and Jean V. Murphy

Companies must develop a strategy to take advantage of trading exchanges, analysts say, but caution is required since most of these new entities are not expected to survive. Analysts offer tips for separating e-marketplace hype from reality.

The movement of business-to-business (B2B) commerce to the internet and trading exchanges continues to be the hottest trend in supply-chain management. The latest forecast from AMR Research of Cambridge, Mass., puts the value of internet-based B2B transactions by companies doing business in the United States at $5.7tr by 2004, or 29 percent of all commercial transactions.

But this figure itself is not important, AMR says in its April forecast. What is important, "is the indication that B2B e-commerce has incredible momentum and will be adopted at a faster rate than many companies realize."

Trading exchanges will be "the single most important catalyst of change," AMR says, predicting that more than half of B2B transactions, an estimated $3tr, will flow through exchanges, or e-marketplaces, by 2004. For some industries the percentage will be much higher.

"Those companies that are not assessing the potential transformative effect of exchanges are risking extinction," warns the research firm.

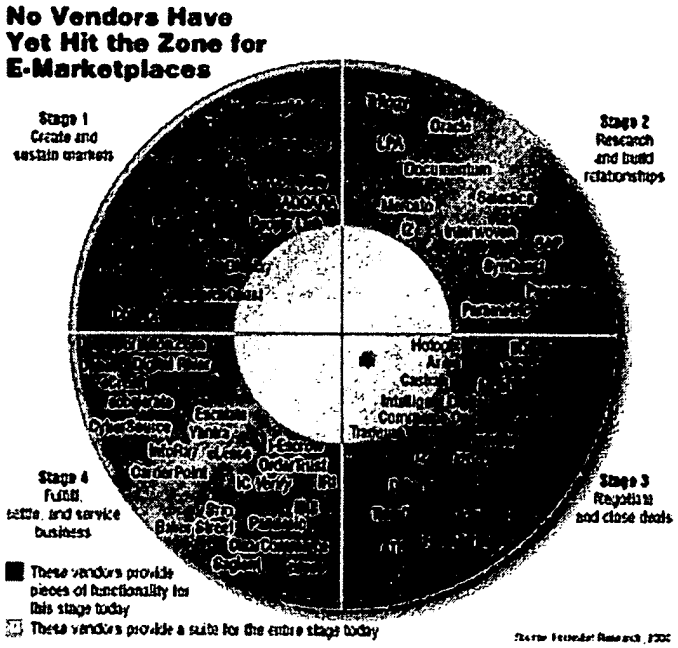
Winning marketplaces will move toward a collaborative commerce e-business model that will allow trading partners to serve as virtual collaborators across a wide array of business processes, the GartnerGroup says.

Sorting out all the hype on the phenomenon of trading exchanges is no easy task, however. There is clearly huge potential but, to date, little real experience. Only 36 percent of 50 leading e-marketplaces are closing more than 100 sales per month, according to a new survey by Forrester Research of Cambridge. Forrester estimates that e-marketplace trade will be about half of AMR's projections by 2004, or $1.4tr. While marketplaces have great plans to add functionality, Forrester says that today most are "loosely constructed and barely

serving the needs of customers."

There also is something of a gold-rush mentality surrounding the launching of exchanges. Dan Miklovic, who leads ERP research for the GartnerGroup, a Stamford, Conn.-based information-technology research firm, says this phenomenon was demonstrated by the recent announcement that Boeing, Lockheed Martin, BAE Systems and Raytheon plan to form a trading exchange for the aerospace industry using Commerce One's portal solution. That announcement, which noted that these companies have signed a letter of intent to form a marketplace, was "almost totally content free," he says.

"Fear and greed are forcing many companies into trading exchanges," notes Charles Gerlach, director of eBusiness strategy at Mainspring, another Cambridge-based market research firm. "They are afraid that competitors will use these systems to gain a competitive edge, and they see the huge numbers exchanges have gained on Wall Street and want to grab a piece of the action. Many have not yet examined what the exchanges will deliver to them from a business perspective."



Making that determination and deciding how to incorporate trading exchanges into a business strategy is a challenge. For one thing, many of today's exchanges aren't expected to survive. GartnerGroup predicts that by 2002 there will be more than 3,000 B2B e-commerce marketplaces, but only 5 percent of these will succeed. Fifteen percent will be acquired or merged and 80 percent will fail. Similarly, AMR says that in the next 18 months the number of trading exchanges will be whittled down to two or three within each industry, as a result of bankruptcy, mergers and acquisitions.

Keys to Survival

These analysts point to a number of key issues that they believe will determine which independent trading exchanges (ITEs) will survive.

One is integration capability. Supporting electronic data interchange and flat file transfers will be insufficient to attract the large, high-volume customers that exchanges need, says AMR. Standardized integration into the leading supply chain management, customer relationship

management and enterprise resource planning systems is a must. "ITEs that don't offer integration capability will wither on the vine this year," according to AMR's recent report, Evaluating the Independent Trading Exchanges.

Liquidity also is important. ITEs are finding it difficult to drive business, AMR says, a fact that will accelerate consolidation. The need for liquidity is forcing independent exchanges to re-evaluate their neutral stance. "They are increasingly accepting investments and partnering with established brick-and-mortar companies," says AMR. "Partnering with an established company provides trading exchanges not only with funds and validity but, more importantly, transaction volume. Because of this pressure, ITEs are giving away the farm to anchor customers."

Supplier reaction is another issue impacting the way exchanges evolve. Many suppliers, says AMR, are balking at ITEs because they see the bulk of transaction fees being shifted to them as buyers resist paying for the privilege of purchasing. Suppliers also are concerned about putting the new internet middlemen between them and their customers. "Suppliers do not want services and products to become commodities," says AMR, "and they have concerns about maintaining brand through an ITE."

As a result, two things will happen in 2000, AMR says. ITEs will begin to offer suppliers a better value proposition by providing logistics, materials handling and customer service capabilities, either in-house or through partnerships. And suppliers will themselves launch exchanges on their own terms.

GartnerGroup says that successful e-marketplaces inevitably will lean toward acting in the interest of the buyer. They will have fewer exclusive supplier relationships and become more "source neutral," using an auction pricing model. A key value-add may be providing editorial content and ratings on suppliers, something suppliers do not much like, says Gartner. "The key will be for e-market makers to brand themselves as buyer advocates while continuing to provide value to sellers. This may entail offering preferred placement to sellers, or it may involve sharing customer data ... This will emerge as the market's 'sweet spot' in the next five years."

Over time, says Gartner, "sellers should expect e-market makers to offer strong fincial viability, quality of product information and a variety of pricing options, as well as a clear data-integration stragegy. Real-time access to customer behavioral data and demographic information will remain a differentiator, as will service and support."

Forrester Research also believes data will prove the key marketplace differentiator, noting that a great opportunity awaits "as high-value information vaults fill up." Neutral e-marketplaces, says Forrester, "should provide data like supplier performance and price history to inform buyers during negotiations. Advanced sites will gain attention with new market-specific indexes — like ticker-tape market pricing or daily inventory levels — or through rich market lists that can be used for blind marketing campaigns."

"What we've found is that trading exchanges have tremendous potential benefits, but a company has to expend a

Winning marketplaces will move toward a collaborative commerce business model that will go far beyond the facilitation of transactions, Gartner says. The research firm's vision of c-

lot of grunt work to fully exploit them."
— Tim Lambeth of VF Corp.

commerce differs from e-commerce in a number of ways. A recent company presentation explains: "E-commerce is designed to construct a virtual link between a pre-defined community of trading partners for the purpose of buying and selling goods and services. Content is generally confined to Web catalogs of finished goods, and collaboration mechanisms center on exchanging messages or purchasing transactions. C-commerce provides a much deeper and richer form of B2B interaction designed to allow trading partners to serve as virtual collaborators across a wide array of business processes. Thus, a c-commerce framework acts as a virtual conduit for connecting information repositories, business applications and business processes, allowing companies to exploit opportunities in the internet-connected economy more easily."

Bruce A. Bond, Gartner vice president, puts it more simply: "E-commerce is about buying and selling," he says. "C-commerce is about sharing intellectual capital."

"It is the next opportunity for corporations to gain competitive advantage," he adds. "With c-commerce, many mission-critical business processes such as sourcing, product design and production will be transformed."

The evolution to c-commerce will occur over the next five years, Gartner predicts. "Currently there are few examples of c-commerce implementations and marketplaces," it says. "By 2004, however, c-commerce will eclipse e-commerce as the dominant, mission-critical e-business model."

Whatever vision proves to be correct when 2004 arrives, it is clear that there is much work to be done in the meantime by marketplaces, vendors and within companies.

New applications must be developed if e-marketplaces are to achieve their goals, says Forrester's Stacie McCullough in a new report, eMarketplace Hype, Apps Realities.Until that happens, she says, "e-marketplaces are stuck in commerce kindergarten."

E-marketplaces, according to the report, need software applications that will enable them to help participants conduct business through all the stages of the trading life cycle — from product discovery to trade dispute settlement. This requires sites to support pre-sale activities like buyer-seller profiling as well as post-order customer services like shipment tracking. To deliver a scalable solution, marketplaces must combine services with packaged applications, the report says, but most currently available software is not designed to support the many-to-many market structure demanded by net markets.

Many vendors, including supply-chain specialists i2 Technologies, Manugistics, SynQuest and Yantra, provide pieces of needed functionality, but none yet has "hit the zone," Forrester says. As a result, vendors are pursuing a flurry of partnerships and acquisitions, such as i2's recent partnership with Ariba and IBM and its acquisition of Aspect Development and SupplyBase.

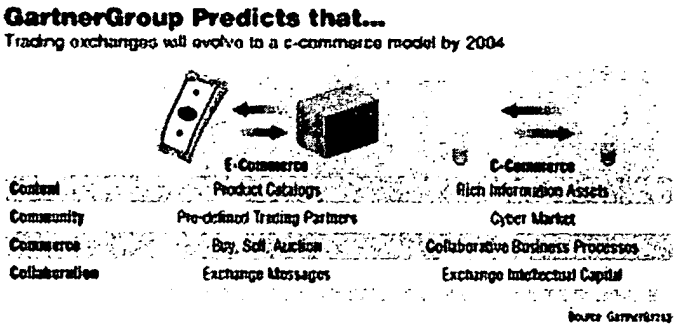
Internal Readiness
To take full advantage of the potential of trading exchanges, businesses that want to participate will have to step up implementation

of supply-chain applications like advanced planning and scheduling, available-to-promise, vendor-managed inventories and collaboration, says AMR.

While leading companies are far along in this process, most others have barely begun.

"It's only been recently that companies have moved away from home-grown legacy systems to integrated ERP packages," says Chad Quinn, vice president of eBusiness strategy at Manugistics Group Inc., a Rockville, Md.-based supply-chain software vendor. "Now that they are wrapping up that work, they are looking to integrate their supply chains."

VF Corp., Greensboro, N.C., represents a typical case. "Four years ago, management decided to invest in new technology thinking it would improve our internal and external business processes," says Tim Lambeth, vice president of global processes at the firm. The company installed ERP software from SAP America Inc., Philadelphia, and has begun implementing i2's supply-chain software and Logility Software's demand forecasting tools.



As these applications come online, they have to be tied together so data will flow among them and VF hired IBM Global Services to perform that work. The apparel company, which handles 250,000 SKUs daily, expects to tie all of the applications used in its jeans business together by the end of the third quarter and then focus on other product lines, such as intimate apparel.

This best-of-breed approach will continue to be the rule for most companies, says Gartner's Bond. That makes integration crucial, especially when two or more trading partners, each with varying applications, are attempting to interact. For new application purchases, this is less of a problem since most now are equipped with application program interfaces. By 2003, Gartner says, more than 90 percent of packaged business software will offer APIs capable of exchanging documents in XML format.

Trading exchanges also are improving their integration capabilities by partnering with Enterprise Application Integration vendors and system integrators.

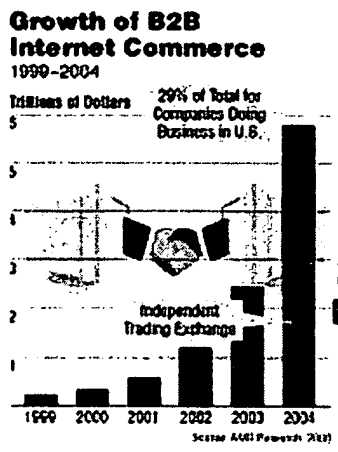
Standards eventually will be adopted that will make integration much easier, but not in the near term. "Partly, this is because it will be years before enterprises completely replace their older applications with systems that use the standard that is adopted," says Gartner's Ross Altman, research director. "Mostly, this will be because too many standards are being proposed, and so it is highly unlikely that all

enterprises and vendors will agree on a single standard in the next five years."

Some companies avoid much of the integration problem by purchasing a full suite of applications from one provider. That was the route taken by Wickes Furniture when it decided on a supply-chain overhaul. "Because our supply chain was largely manual, our salespersons were not able to tell customers when a product would arrive as they placed their orders," says Ken Maher, vice president, controller, and corporate secretary at Wickes Furniture. "Management decided we had to change our systems because some customers would walk out the door to competitors who could give them a firm delivery date."

Wickes examined deploying a new ERP system but instead purchased Logility's Voyager supply-chain software. "We wanted to improve a customer's buying experience and supply-chain systems have a much bigger impact on that than ERP software," says Maher.

The company plans to deploy Logility's full suite: Event Planning, Demand Planning, Inventory Planning, Replenishment Planning, Demand Chain Voyager, Transportation Planning and Management, Voyager XPS and Voyager XES.



The latter two products will support Wickes's participation in Logility's hosted trading community. Wickes, which operates 32 showrooms, will use Logility's i-Community to improve communications with its manufacturers and transportation providers. It initially plans to connect with supplier Ashley Furniture of Arcadia, Wis.

[redacted] which also includes HightechMatrix.com and FreightMatrix.com

Anchor tenant VF will initially focus on indirect goods suppliers, but plans eventually to expand its marketplace activities to include direct goods. "Currently, we do not have a good picture of where needed materials are in the supply chain," says VF's Lambeth. "Once the exchange is in place, we'll have a complete view into our supply chain and be able to fulfill new orders more quickly. We'll also lower inventory and reduce product obsolescence.

"What we've found is that trading exchanges have tremendous potential benefits, but a company has to expend a lot of grunt work to fully exploit them," he adds.

As trading exchanges mature and new software applications become available, they will begin to add more functionality and to offer valuable market information. Companies are cautioned not to wait until all the pieces are in place before developing an e-marketplace strategy, however, particularly in industries that are moving quickest to internet-based commerce. Procrastinators may well be run down in this fast-moving race.

To help companies in this process, AMR Research has developed a number of recommendations summarized below.

For Buyers:

• When evaluating exchanges, demand strong process support and integration. Select an ITE that minimally integrates with an organization's existing workflow and requisitioning processes or provides that capability. For an ITE to be an integrated component of the supply chain it must address all four phases of the procurement process: request, buy, supply and remit.

• Bargain hard on prices, setup fees and integration services. Equity is not out of the question.

• Understand the financial backers and operators of the ITE. Many ITEs solicited funding from established brick-and-mortar companies; some of those organizations may be competing firms. If a competitor owns an equity stake, does it have access to the transaction data?

• Determine how the exchange handles third-party services, such as supply-chain execution. Who is ultimately responsible for getting the product to you?

• Protect your suppliers. If you have spent the last 10 years rationalizing your supplier base, carefully consider how an ITE will impact your partners. It is crucial to understand that if an ITE gets revenue from your suppliers, they are going to be asked to deliver the same level of services at a reduced profit.

For Suppliers:

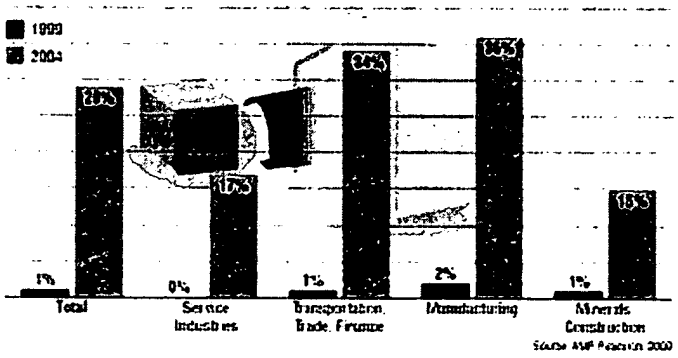
• Bargain hard on transaction fees, service fees, contract terms and margins. Contrary to the popular press, ITEs need you more than you need them.

• Develop a substantial content strategy. It will enable your organization to support multiple exchanges.

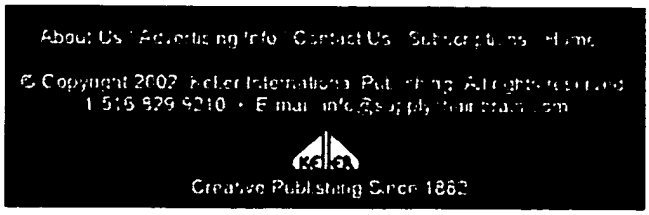
• Do not get blindsided by a significant customer that selects an ITE as its sole e-commerce platform. Proactively approach customers about their plans and ITEs within your industry.

Internet Commerce Penetration Estimates

1999 vs. 2004



Back to top


About Us | Advertising Info | Contact Us | Subscriptions | Home
© Copyright 2002 Keller International Publishing. All rights reserved.
1 516 929 9210 • E mail: info@supplychainbrain.com
Creative Publishing Since 1862

DNJ ONLINE

Newsreel
Products & Services
Web Watch
Software Update
Resource Directory
Events Diary
Articles
The Magazine
Subscribe
Contact Us

Search DNJ Online

Head to Head

At your service

July 98 - Matt Nicholson talks pipelines, components and Web security with Arnold Blinn, Microsoft's Commerce Server architect, at the recent European Internet Commerce Developer Conference in Amsterdam.

Matt: What actually is Site Server?

Arnold: Site Server is a collection of different things. I'm often asked whether Site Server is a platform or an application. My answer to this question is 'Yes'. The reason I say that is because, in a world where everybody is rolling their own Web site, there's actually a very thin line between an application and a platform.

There's a number of COM objects in Site Server for building Web sites, and there's some tools, such as the analysis tool. There's a lot of tools in the personalisation and membership system for accessing user properties and personalising pages for users.

Then there's the components in the Commerce Edition [previously known as the Enterprise Edition]. The pipelines are definitely the core of this edition but there's also things like the Ad Server; the Auction component for implementing Internet auctions; the Intelligent Cross-sell component and other promotional components. There are also objects which help you build multi-lingual sites that support multiple countries at the same time. So there's a lot of things beyond the pipeline.

Matt: Are these components all part of the commercial product or are they coming later?

Arnold: Everything I mentioned is in the product, with the exception of the Auction component. The Auction component is something we're working on for download later this summer, and an example site that you can download into your Commerce

installation. We'll also release some white papers and additional examples around that time.

Matt: Is the Auction component a COM object?

Arnold: Yes. Like the Ad Server there's a COM object and then there's some tools and management pages to manage how the COM object works. That's pretty universal across a lot of the objects - you have the run-time part and then you have the configuration and administration part.

Matt: I got the impression that many of the administrative tools are ASPs [Active Server Pages].

Arnold: A fair number are, but not all. Let me do a taxonomy of our users. There's the NT administrator who configures SQL Server and creates disk partitions and the like. We have an MMC [Microsoft Management Console] snap-in for him that creates site foundations, database devices and so on. Then there's the site author, who might be a Webmaster. He uses Visual InterDev, FrontPage, the Pipeline Editor and DTCs [Design Time Controls].

The next type of guy is the store manager - somebody who sets up promotions, tax rates, shipping rates and so forth. He might run the Ad Manager application to set up ad campaigns, the Promotion Manager to set up discounts and promotions, or the Auction Manager to set up auctions. But of course while the store manager creates ad campaigns, the site author is creating pages that call the Ad Server COM object.

You have to keep in mind that sometimes there can be one guy doing all three of these jobs, and sometimes there's ten guys doing each job. In the case of the manager you might have one guy who sets up ad campaigns, another who changes product prices, and another who changes shipping rates. It depends on the size of the site and the way they do their business.

Matt: Where configuration and management is implemented as ASPs, you can presumably administer these remotely using Internet Explorer?

Arnold: Yes, that's correct.

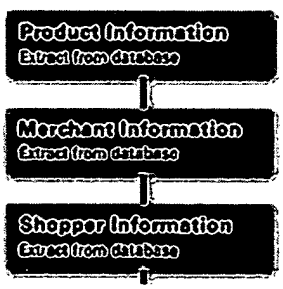
So, to get back to your earlier question, "What is Site Server Commerce Edition?" From a developer point of view, Site Server is a bunch of services that are exposed through MMC in the same way as a directory server is exposed through an MMC; it's authoring tools like the pipeline editor; and it's ASP applications such as Ad Manager or Site Wizard, which allows novices to create complete sites.

Matt: If we take one of these components - say the Ad Manager - is that written in C++?

Arnold: Internally it's all implemented in C++. That's how we write our COM objects. It's a fairly closed object. Now in some situations, like the pipeline, we also say, "You can build your own plug-ins for this framework." Even then you're building COM objects.

We want people to build pipeline components so we ship example source code in the box in Java, Visual Basic and C++, but it's not across the board. We don't expect people to build another Ad Server. The Ad Server's the Ad Server: here's the COM object, here are the methods - call it.

Matt: Could you describe how the pipeline works?



Arnold: The pipeline is the meat of the system. A pipeline is a business process that we model in a framework. A framework consists of stages. Stages are implemented by components, and the components operate on a business object.

An example of a business process is 'compute the amount of the order'. The framework consists of

stages. In this example the stages are things like compute the shipping, the line item prices, the handling, the tax, the total. The stages are implemented by components, so compute shipping based on the rates from Federal Express, or compute the rate based on UPS, could be different plug-ins for the shipping stage.

The components themselves operate on this thing called a business object. In the case of Commerce Server, a business object comprises two lower level objects which we call a 'dictionary' and a 'simple list'. They're both very simple objects.

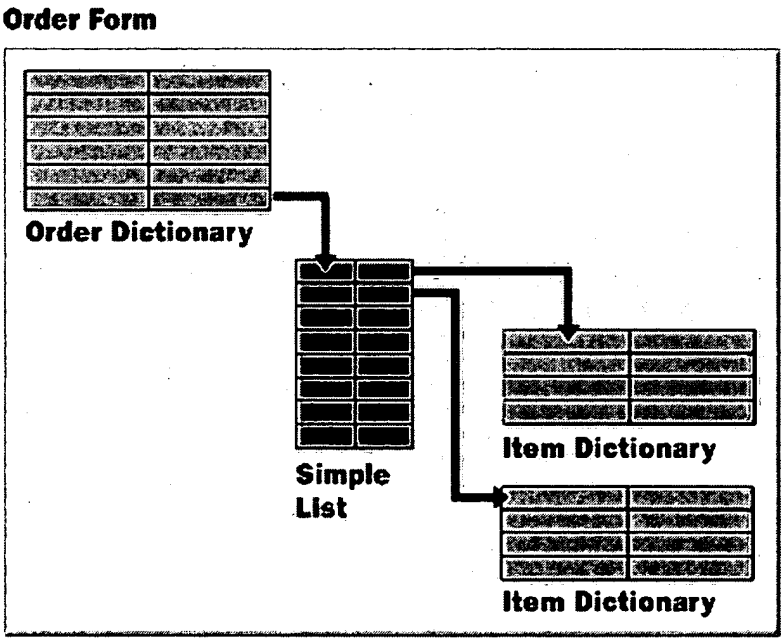
A dictionary is a collection of key value pairs where the keys are strings and the values are variants, which means they can be strings, numbers, integers, floating points or even other objects. The simple list is similar to an array but it's just a list of variants. The difference between a simple list and an array is that the items in the simple list can be different data types, so you can have a string followed by a number and then an object.

A further difference is that both the dictionary and the simple list objects know how to save themselves persistently. They can be saved into a database, into a file, they can be saved as binary data or even as an XML representation.

The business object depends on the pipeline, but in the case of the order pipeline it's something called an order form. This is simply a dictionary of key value pairs where one of the key values is called Items and points to a simple list of other dictionaries which each represent an item in the order.

The components operate on the business object. When the pipeline runs, all the components do is read values from and write values back to the order form object - either to the order dictionary or to the item dictionary.

So the pipeline is simply a list of components that get executed one after the other. The order form gets passed to the first component, which reads and writes values to it; then to the second component, which reads and writes values back to it; then to the third one, and so on.



Each component writes values to the order form that later components in the pipeline expect to be there. For example, for each line item, the Product Info stage in the pipeline looks up information about the product and writes data back into the line item dictionaries. If you have a shipping component later in the pipeline, it will expect there to be a field called Weight because it needs the weight to do its work, so the Product Info component had better write a value called Weight into the item.

Matt: What happens if it doesn't?

Arnold: The developer would get an error that would tell him: "You tried to read a value called Weight from the product and it's not there." So the developer would say: "Oh, I have shipping component that needs Weight, so I'd better write it in an earlier stage."

So that's how the pipeline works. The value of the pipeline is that it's a very flexible architecture as you can define the components that implement each stage. In fact there's an advanced mode on the pipe editor tool where you can even add and delete your own stages from the process.

You might modify the business process and say, "I don't have shipping, so I'm going to delete the shipping stage." People get all excited when they hear about this, but the problem is that later stages might expect there to be a value for shipping.

The total stage, which comes after shipping, sets the order total to equal the sub total plus shipping plus handling plus tax. If I go ahead and delete the shipping stage, there's not going to be a value for shipping because nobody wrote it, which means the component would fail.

Of course I could write a new component that just does sub total plus handling plus tax. The system would still work and now I have a pipeline that doesn't need the shipping stage.

Matt: Couldn't you also create a shipping object which simply sets shipping to zero?

Arnold: Exactly. And in fact by default that's what an 'empty' shipping stage would do, so you wouldn't have to delete it in the first place.

That's not to say that the advanced mode isn't useful. The flexible stages and the advanced mode and being able to plug in arbitrary components is all well and fine and interesting, but the real power of the system comes in what the components themselves are reading and writing to the pipeline. We've defined the stages; we've defined what values they write. But we want third parties to write the pipeline components.

Matt: So you're trying to create a standard here?

Arnold: That's correct – in a manner of speaking. It's a standard for component vendors to work together in an integrated environment. We are enabling third parties to write functionality specific to their business. Yes, they could do that without our help. They could expose their functionality as COM objects, but it would be a pain in the neck because you'd have to talk to one API for one component, and another API for another. It would be difficult because nothing would work together.

We're creating a standard where a tax vendor, a shipping vendor and a payment vendor all have a uniform API and a single, uniform tool to configure their system to work together.

Matt: What about adding new stages?

Arnold: Go for it man! That's better than deleting!

Matt: Because components which fall further down the chain aren't going to be affected by new values - they're just going to ignore it.

Arnold: That's correct. And some of the stages that we've defined aren't dependent on other stages. The inventory check stage is a good example. Then there's the merchant info stage which nobody ever uses. I should just delete it from our example stores - nobody's building components for it, nobody's using it, so just delete it!

Matt: Tell me about other pipelines.

Arnold: The Commerce Interchange Pipeline [CIP] is different, and actually has two pipelines. The first is the send side and the second is the receive side.

On the send side we've defined the stages as map the data, format a header, sign, encrypt, audit and send. This is the business process which takes a business object and sends it to another trading partner. The business object can be pretty much anything, although typically it's an order form. But it could be an OLE DB

where you're basically re-inventing programming. And if I'm re-inventing programming, why don't I just use something that's really good at programming, like Visual Basic Script? Why not just call the components from VBScript?

So we have one pipeline component that just executes a script. It's a script host, just like ASPs is a script host. It creates an instance of the scripting engine. You click on the thing and it has a little window where you enter your VBScript. And just like ASPs, there are intrinsics available to that script, such as the root object that is being passed from stage to stage in the pipeline. So you can write your script to say, "If OrderForm.GiftMessage is not null, then do something."

Matt: A big concern in Europe is cross-currency purchasing. How does Site Server handle that?

Arnold: The Commerce Edition has some objects in the server-side architecture that allow for the arbitrary conversion and display of currencies from any Windows NT supported locale that's on the machine. In fact we store all monies in whole numbers - so in the United States we store currency as pennies, we don't store it as US dollars. A dollar is a hundred in Site Server. The reason we do that is because none of the components in the order process pipeline deals with fractions. They just deal with integers to avoid rounding errors.

So what about display? The currency is formatted when the page is generated for the shopper. What the system does have is a number of features allowing the display of currency, date, time in different formats from the same Web site.

Security on the Web

Matt: We're talking here about using a credit card over the Internet. The impression I get is that it's actually more secure now than giving your credit card over a telephone.

Arnold: I thought you were going to say, "The impression I get is that it's more secure now than it was two years ago." That would be funny because it's hardly changed! The paranoia about security is largely the press's fault. But to answer your question, it's typically safer than handing your credit card to a waiter at a restaurant.

The danger of credit cards on the Web is not the transmission of credit card details across the Internet. SSL is really secure in the US. In Europe it's OK. I won't comment on the political situation that prevents you from getting stronger encryption here, but there's other payment protocols that solve that problem. I've never heard of anybody picking a credit card up off a wire in the clear, let alone when it's been encrypted. It's ludicrous to think of somebody doing that, other than for the pure sport.

The security problem is frankly once the credit card is stored on the server at the store, because then the hacker can go in and steal thousands of credit cards by breaking into one server without having to do any wire sniffing. There have been a lot of documented cases of that type of fraud. That's why Commerce server does not save credit card numbers on the server by default.

Now of course a real world application would need to save the card number if they wanted to get payment later, and we have a pipeline component that will encrypt the number. You can take that encrypt component and plug it into the OPP into the accept stage. Unfortunately you still have this limitation of 40 bit encryption, but someone in Europe could write a component with stronger encryption.

Top of page
Back to Articles

WWRE Overview

In March 2000, 17 international retailers founded the WorldWide Retail Exchange (WWRE) to enable part retailers and manufacturers to simplify, rationalize, and automate supply chain processes, thereby elimin inefficiencies in the supply chain. Today, the WWRE is the premier Internet-based business-to-business (in the retail e-marketplace. Utilizing the most sophisticated Internet technology available, the WWRE ena and manufacturers in the food, general merchandise, textile/home, and drugstore sectors to substantiall across product development, e-Procurement, and supply chain processes. Current membership consists c industry leaders from around the world with combined revenue of over U.S. $900 billion.

To date, the WWRE has saved its members over $1 billion. The Exchange operates as an open, independ company that generates benefits for its members a (Retailers and Manufacturers), and ultimately the cor end, the WWRE is run as a private company with no plans of going public. Rather, the WWRE is intent or effort and resources on bringing value to its customers and improving supply chain efficiency within the r

Founding Principles

The following six principles are among the principles that guide the WWRE's development and growth:

- Openness
- Commitment to utilizing the best available technology
- Focus on improving efficiency and lowering costs for the retail industry
- Operation as a neutral company
- Equivalent fee structures for all participants
- Confidentiality of transaction information

Value Proposition

The WWRE will continue to be the premier integrated worldwide exchange community for retailers and m by improving efficiency and lowering costs throughout the supply chain. There are seven key ways for m realize value:

- Low-cost product offerings that are robust, scaleable, integrated, and fully supported
- Shared technology investments and outsourced assets
- Ability to access a global membership community and network with other retailers/manufacturer
- Value-added services from a trusted source, at competitive costs
- Participation in collaborative activities
- Complex transactions and interactions made easy through automation
- Standard setting benefits for all B2B activities

Products & Services

The WWRE offers a full-scope procurement platform that allows participants to conduct a full range of e- transactions. Core products and services include Negotiations & Auctions, Collaborative Planner; WorldWi Management, eProcurement, WorldWide Design Planning & Management, and WorldWide Trade Logistics

Management Team

Colin Dyer, Chief Executive Officer

Member Statistics

62 Members

Robert Heaton, Chief Financial Officer
Ramana Palepu, Chief Technology Officer
Sally Herbert, Chief Operating Officer
Nick Parnaby, Chief Marketing Officer
Richard Harris, General Counsel

Stores in more than 13(
Over U.S. $900 billion ir
Over 5,000,000 employ

Global Headquarters*

WWRE
625 North Washington Street
Alexandria, VA 22314
U.S.A.
www.wwre.org
phone: 703.234.5100
fax: 703.234.5200

* Representatives located in the Asia-Pacific (Tokyo) and Europe (Paris).

Awards and Accolades

- Forbes - Listed in September 10, 2001 special Business to Business issue of Forbes as a Best of t honoree in the retailing category.
- CIO-100 - Placed on CIO magazine's "CIO-100" list, which recognizes organizations around the v in positive business performance through innovative practices and products. This year the list of companies includes those that have "demonstrated innovation to improve products, services, an with partners and clients."

WWRE Members and Participants to Date:

AEON Co., Ltd
Ahold
Albertson's
Auchan
Best Buy
The Boots Company
C&A Europe
Campbell Soup Company
Casino
Controladora Comercial Mexicana S.A. de C.V. GDS
Coop Italia
Coop
Cora
CVS/pharmacy
Dairy Farm International
Dansk Supermarked Gruppen
Delhaize Group
Dixons Group plc
Edeka
El Corte Ingles
Galeries Lafayette
Gap Inc.
GlaxoSmithKline plc
Giant Eagle
H.E. Butt Grocery Company
Hy-Vee Inc.
Izumiya Co.
JCPenney
Kesko
Kingfisher
Kmart Corporation
Laurus
Longs Drugs
Lotte Group
Makro Asia
MARKANT
Marks & Spencer
Meijer, Inc.
Otto Versand
Publix Super Markets
RadioShack Corporation
REWE
Rite Aid Corporation
Safeway Inc.
Safeway plc
SCA Hygiene Products
Schering-Plough HealthCare Product
Schlecker
Seibu Department Stores Ltd.
ShopKo Stores, Inc.
Sobeys Inc.
SUPERVALU INC.
Target Corporation
Tengelmann Group
Tesco
Toys R Us
Wakefern Food Co.
Walgreen Co.
Wegmans Food Markets, Inc.
Winn-Dixie Stores, Inc.
Woolworths
Wyeth Consumer Healthcare

"One of the particular things that we liked about the i2 solution was how each company had the flexibility to decide which pieces they would use. It addresses so many different business needs that, regardless of your process, you can use the solution."

—Sylvie Weeks, Senior Director of Sourcing and Procurement, **WorldWide Retail Exchange**

Industry
Retail

i2 Success Story **#236**

Delivering 24/7 Solutions for Members of the WorldWide Retail Exchange

When leading global retailers decided to collaborate in a Web-enabled marketplace called the WorldWide Retail Exchange, they knew that they would need flexible and robust solutions to serve as the platform. Looking to leverage technology and reduce supply chain costs for its members, WWRE implemented i2 Procurement™ and i2 Negotiate.™ Since partnering with i2, WWRE has increased member satisfaction and retentions and has remained competitive by attracting new members.

Most retail companies strive to streamline procurement and supply chain processes to create new business efficiencies and drive revenue growth. However, the continually evolving nature of the retail industry and its customers makes this a daunting task for an individual retailer in this global economy.

In 2000, a group of 17 leading global retailers realized that the most efficient way to leverage technology and reduce supply chain costs would be to collaborate in a Web-enabled exchange. The companies joined together to form the WorldWide Retail Exchange (WWRE) with the goal of facilitating and simplifying trade among retailers, suppliers, partners, and distributors.

WWRE's founders knew they needed a technology foundation that could differentiate them from their competitors and provide for the diverse needs of the member companies.

Why i2?

WWRE sought to identify a provider that could offer flexible, reliable solutions that would be scalable to the needs of numerous companies.

With thousands of users across the world, WWRE needed a solution that could handle a 24-hour, seven-day-a-week operation while providing a wide enough range of capabilities to address the diverse business models of the member companies.

WWRE found the capabilities it was looking for in i2 Procurement and i2 Negotiate.

"We needed a solution that had significantly more features and functions than any one company would use because each company we deal with has different issues," said Sylvie Weeks, Senior Director of Sourcing and Procurement for WWRE. "One of the particular things that we liked about the i2 solution was how each company had the flexibility to decide which pieces they would use. It addresses so many different business needs that, regardless of your process, you can use the solution."

i2's Contribution

i2 Procurement provides end-to-end purchasing lifecycle management, from requisitioning to purchase-order creation to settlement. It also delivers a workflow-driven, self-service interface that enables users to generate requisitions for multiple products from various suppliers. For WWRE, i2 Procurement enables member companies to build business rules and workflow patterns that are customized for each organization.

Challenges
- Establish platform for Web-enabled exchange
- Identify reliable and robust solutions to handle members' diverse operations
- Ensure member satisfaction through reduced supply chain costs

Solutions
- Build unique business rules for members
- Accelerate purchasing process
- Eliminate language and currency barriers

Results
- Exceeded expectations for numbers of users and catalogs
- Increased member satisfaction and retention
- Attracted new members by offering unique capabilities

"You can get an auction solution pretty much anywhere right now, but we want to provide our members with a solution that they value to the point that they don't ever question leaving – and for us i2 Procurement is definitely that solution," Weeks said. "Renewal never becomes a question because we are providing them with a solution that meets their needs, has all of their feature functions, and has been so successful that they now have hundreds of users, and they can't think about what they would do without us."

"One of the things that we liked about i2 Procurement was that every company could individually design its workflow needs or in some cases simply turn that feature off if it didn't use it," Weeks said.

i2 Negotiate enables purchasing organizations to negotiate the best terms for an optimal supply base by automating the quote-build process and providing back-end quote analysis functionality. With i2 Negotiate, members of WWRE can accelerate the purchasing process and create strategic supplier relationships – adding efficiency to the supply chain.

"Members can very quickly pull together an RFP or RFI, send it out to a wide variety of suppliers, and then allow the system to consolidate the responses as they monitor the process and respond to questions in real time," Weeks said.

i2 solutions have also enabled the members of WWRE to operate on a global scale by eliminating language and currency barriers. A company simply sets its home language and currency and then shops from any catalog. When an item is selected, the approval process continues with the price automatically appearing in the user's home currency.

WorldWide Retail Exchange's Results

While other exchanges have folded as a result of the dot-com bust, i2 has helped WWRE to remain a viable and competitive Web-enabled marketplace.

WWRE measures results by focusing on the value its solutions create for its members. i2 solutions have proven to be an integral part of that value-creation process and have enabled the company to exceed its growth expectations.

"When we look at things like number of users and number of catalogs, we see that we are actually increasing faster than we expected," Weeks said. "Our year-long goals were increased by 100 percent from the previous year and we are on track to meet those targets."

Although WWRE is in the early stages of its i2 Negotiate implementation, the company is already getting positive feedback from its pilot group. Additionally, several companies who have in-house procurement solutions that do not include the Negotiate capabilities are seeking out the exchange to use this function.

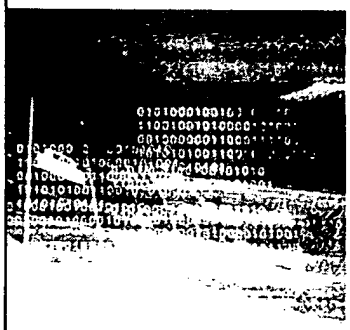
"It has opened up a completely new market to us that we didn't originally anticipate," Weeks said.

Another important aspect of the i2/WWRE relationship is the effect that i2 solutions have on member satisfaction and retention. i2 provides the company with solutions that are unique and robust, allowing WWRE to stay competitive while meeting its members' needs.

"You can get an auction solution pretty much anywhere right now, but we want to provide our members with a solution that they value to the point that they don't ever question leaving – and for us i2 Procurement is definitely that solution," Weeks said. "Renewal never becomes a question because we are providing them with a solution that meets their needs, has all of their feature functions, and has been so successful that they now have hundreds of users, and they can't think about what they would do without us."

i2 Success Story **#236**

The WorldWide Retail Exchange (WWRE) is an Internet-based business-to-business exchange in the retail e-marketplace. WWRE enables retailers and manufacturers in the food, general merchandise, textile/home, and drugstore sectors to substantially reduce costs across product development, e-procurement, and supply chain processes. Headquartered in Alexandria, Va., WWRE's members include more than 60 retail industry leaders from around the world with combined revenue of more than US$900 billion. To date, the WWRE has saved its members more than $917 million.



One i2 Place
11701 Luna Road
Dallas, Texas 75234, USA
Phone 1.877.926.9286
Email info@i2.com
Web **www.i2.com**

Extending Site Server 3.0 Pipelines

By Marco Tabini

Pipelines are one of the most exciting technologies in Site Server, Commerce Edition 3.0 (SSCE from now on). Their main characteristic is that of providing an ordered approach at solving business problems. Generally speaking, pipelines are a good solution whenever you need to perform a task that:

1. Can easily be divided in smaller steps
2. Operates on the same set of data
3. Is composed of operations that must occur in a specific order

The most obvious of the possible processes that can be performed using pipelines is **order processing**, which is at the core of an electronic commerce store. It should be pointed out that order processing—at least as we are using it here—is a rather broad term that encompasses everything from managing the customer's shopping cart to determining the total cost of an order, including any shipping, handling and tax fees. Other possible uses include managing business-to-business communications, such as line-of-business operations and other interchanges

Marco Tabini
Marco Tabini is an electronic commerce specialist based in Toronto, Canada. He provides custom solution for clients who desire to bring their business online, and writes about electronic commerce and Site Server on various publications. He is co-author of "Professional Site Server 3" (Wrox Press, 1999).

A quick introduction to pipeline technology

Most of the presentation, as well as these notes, assume that you are quite familiar with pipeline technology. You should at least know its basics, and have used pipelines in one of your stores. The goal of the brief introduction to pipelines that you are about to read, therefore, is primarily that of creating a common foundation for the more advanced concepts that will be covered later on.

What is a pipeline?

As we have already mentioned, from a conceptual point of view a pipeline is simply a business process broken into its essential components and executed in a specific order.

From a technical point of view, a pipeline is simply a COM object configured to perform certain operations. The exact

number of operations to be performed, their order and their parameters are loaded from a *configuration file* (whose extension is .PCF by default).

Although pipeline objects and pipeline configuration files are obviously two different things, we'll often use "pipeline" as a generic term for referring to a pipeline object which has been assigned a particular configuration file.

A pipeline is composed by an arbitrary number of **stages**. Each stage is simply a way to group operation of a similar nature in a logical fashion. Each operation inside a pipeline is carried out by a COM component that implements particular interface, generally known as a **pipeline component**. Site Server includes a number of built-in components whose function varies from collecting information about the products in a user's shopping cart to encrypting a set of information for a business-to-business communication.

Types of pipelines

There are two types of pipelines built-into SSCE. The **Order Processing Pipeline** is used for managing order processing operations in a business-to-consumer online store, while the **Commerce Interchange Pipeline** is used to handle business-to-business communications. Both these pipelines have their own set of stages and dedicated built-in components.

The Order Processing Pipeline

Three subtypes of pipeline templates belong to the Order Processing Pipeline (OPP). Each of them addresses a particular moment in the store workflow:

- **Product pipeline**

This pipeline is used whenever the user requests information about a particular product in the store's catalogue. It extracts all the appropriate data from the database and applies line-item discount to deliver the most up-to-date information.

- **Plan pipeline**

The plan pipeline is used whenever the store needs to show the user with a subtotal of his or her order. This pipeline includes the calculation of taxes, as well as shipping and handling costs.

- **Payment pipeline**

Finally, the goal of the payment pipeline is to take care of completing a purchase by performing such tasks as processing payment information, updating the store's inventory, initiating any line-of-business procedures, and so on.

A fundamental difference between the product and plan pipelines on one side and the payment pipeline on the other is the fact that the latter requires a transactional environment in order to run. Transactions are handled using Microsoft Transaction Server (MTS) 2.0 and the transactional context is provided directly by the pipeline object that is used to execute the pipeline.

The Commerce Interchange Pipeline

As already mentioned earlier, the Commerce Interchange Pipeline (CIP) is used to perform business-to-business tasks.

These usually entail the transmission of a set of information—called **interchange**—between two computer partners. A **receipt** can optionally be sent by the receiving end to acknowledge that an interchange was successfully received.

Clearly, the format in which the interchange is sent must have been previously agreed upon by the two parties, or it will be incomprehensible to the receiving end. The CIP per se is *protocol agnostic*, in the sense that it does neither favor nor require a particular protocol to be used when transmitting and receiving data. As a matter of fact, one of the great things about pipelines is that they can be extended in an arbitrary way by writing custom components—and therefore you can support whatever format you require to.

There are only two subtypes of CIP pipelines:

- **Transmit pipeline**

This pipeline is used to format the interchange according to the appropriate rules and send it to the recipient.

- **Receive pipeline**

The receive pipeline sites at the other end of the interchange, and is invoked whenever a transmission is received by the server on which it resides.

Transmissions can take place in a variety of ways, including several Internet protocols such as HTTP, SMTP or a DCOM transmission. Pipeline extensions can provide interfaces with custom protocols, such as VANs and frame relays.

Data providers

There are two data providers within a pipeline. The **dictionary** contains the information that the pipeline is supposed to manipulate as a result of its execution. For the OPP, the dictionary is an Orderform object containing all the information required to process an order. For the CIP, it is a Dictionary object containing all the information required to create and transmit an exchange or, for the receive pipeline, the interchange that must decoded and interpreted.

The **context** represents a collection of the support data required by the pipeline to run correctly, and is also a Dictionary component that is created before the pipeline's execution and passed as one of the parameters when the pipeline object is invoked.

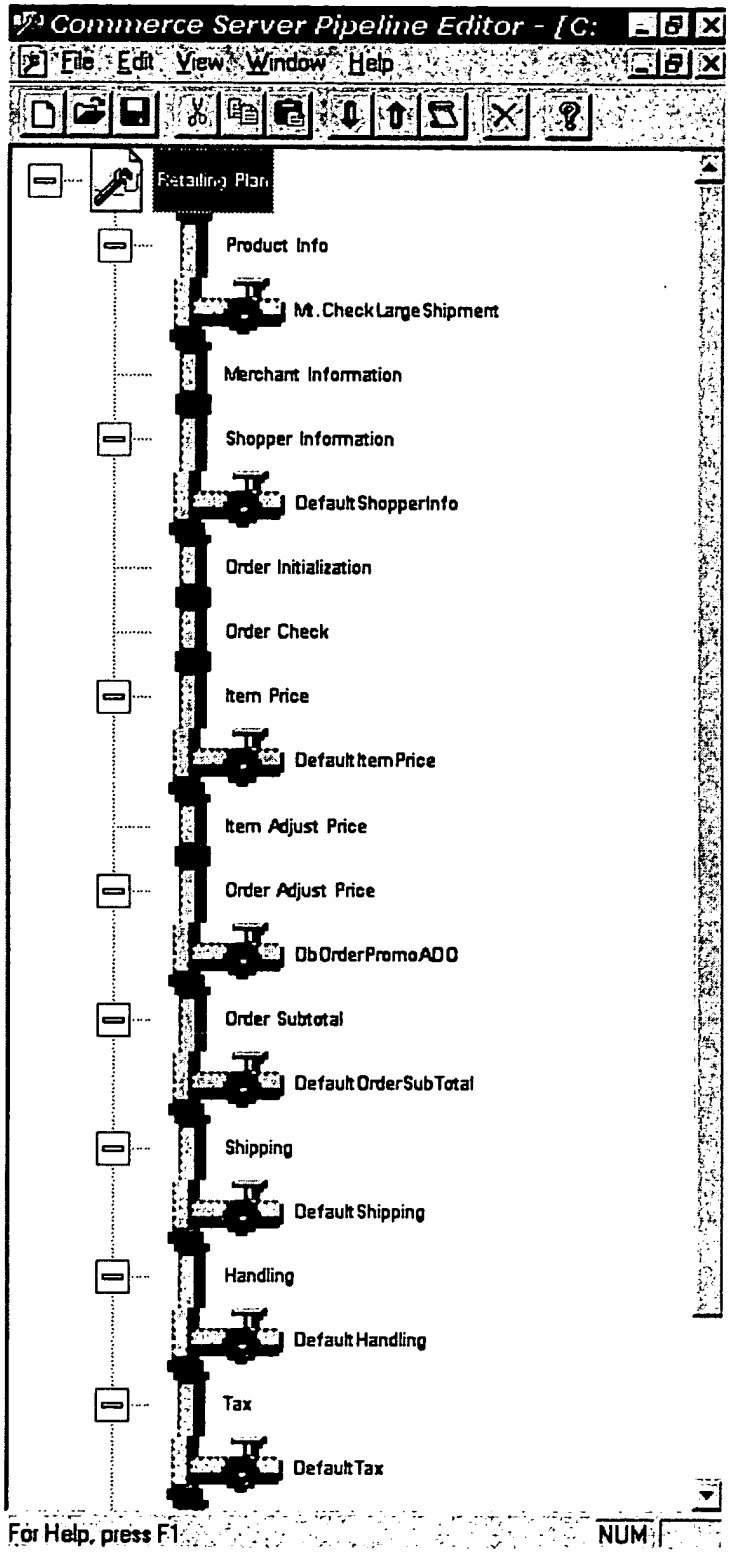
Stored in the context can be a variety of commonly used MSCS objects, such as intrinsic ASP objects (e.g.: Response, Request, and so on), the MessageManager, the QueryMap and others. The context can contain an arbitrary number of objects and other data values of any type that can be simply added as a name/value pair to the Dictionary object.

Editing pipelines

SSCE offers two ways for editing pipeline configuration files. A **Win32** editor can be used if the PCF file is directly accessible from the machine where the editor is running through an NT filesystem or UNC pathnames. The system also includes an **HTTP** editor that can be used through a normal Internet connection (using the appropriate login information).

Generally speaking, the Win32 editor is more powerful than its HTTP counterpart, because it takes full advantage of the Windows GUI and makes it possible to create customized pipelines (which we'll briefly introduce below). However, the

HTTP editor is extremely convenient, since it can be used from anywhere, as long as your server is reachable from the Internet.



The Win32 editor

The Win32 editor is a C++ application that can is installed together with SSCE. It provides a simple and intuitive way to

edit pipelines through the graphical interface that you can see above.

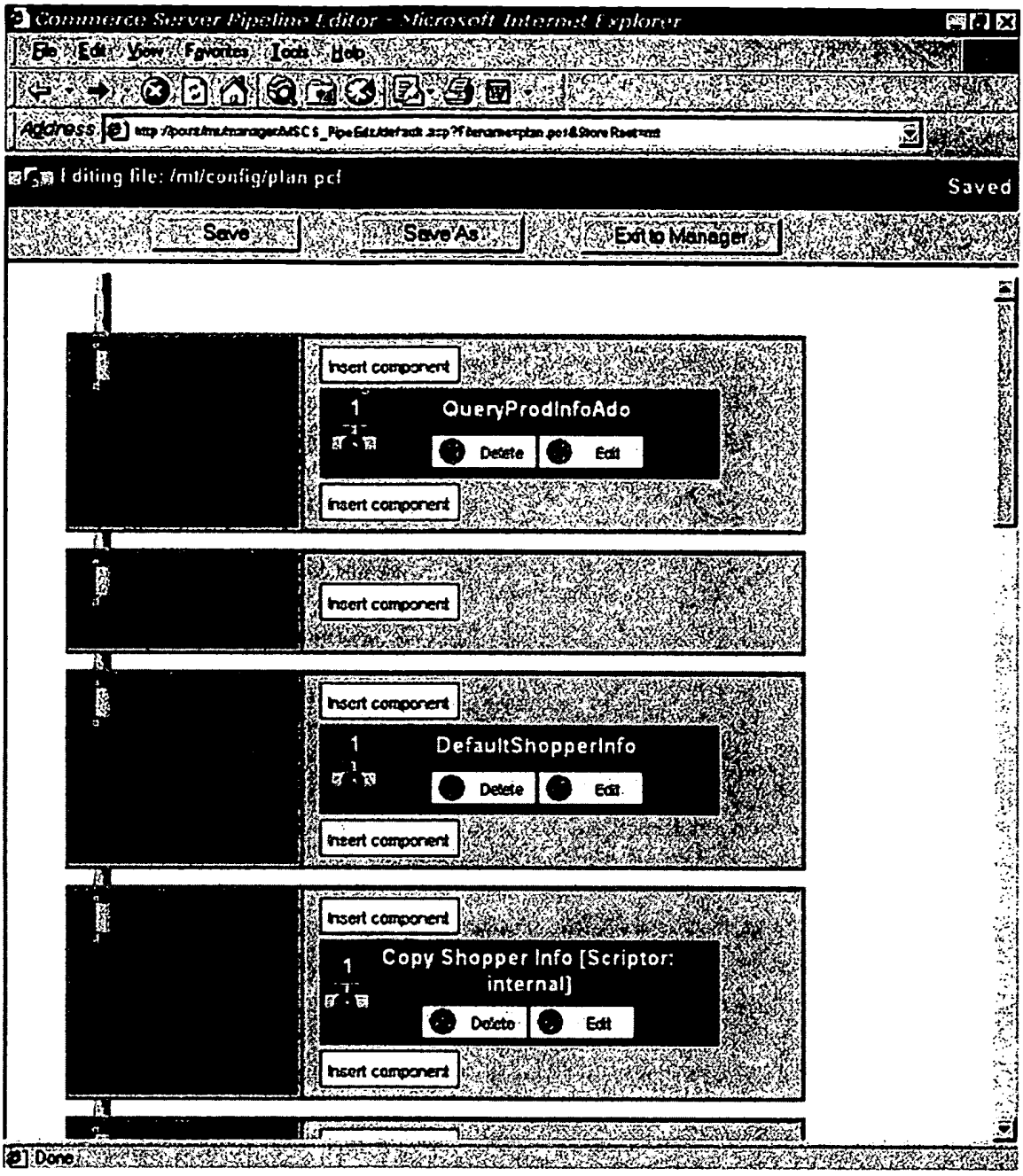
The range of functionality provided by the Win32 editor is very varied. Besides adding and removing components from the pipeline, it is also possible to modify their property and, if the components implement the proper COM interfaces, view the name/value pairs that they access in both the context and the dictionary.

In "Expert" mode, the Win32 editor makes it possible to create custom pipeline that do not necessarily follow the same sequence of operations as their pre-defined counterparts. This allows the developer to create pipeline that address problems beyond order processing and business-to-business interchanges: stages can be moved around, added or deleted altogether, and components can be hidden so that they become "built-in" when the editor runs in normal mode.

The HTTP editor

On the other hand, the HTTP-based editor provides a smaller set of functions. First of all, because the access to COM functionality through an HTTP connection is limited, a special set of ASP pages must be created to let the user edit the properties of a component. Naturally, this means that, if you want to develop a component, and you want it to be fully compatible with all of Site Server's functionality (quite an important requirement, particularly if you want to sell it as an add-on to SSCE), you will have to work out property pages that can work both in their Win32 and HTML form.

In addition, the HTTP editor doesn't have any "expert" mode. This means that it can only be used to edit existing pipelines, and only those pipelines that respond to SSCE's standard specification (that is, traditional OPP and CIP pipelines only).



Pipeline interfaces

When writing a pipeline component, you must implement one or more COM interfaces in order for the pipeline objects and editors to become aware of the fact that your component exists. Even though there are several interfaces that you can implement, only one is absolutely required, and in general you will not necessarily need to use more than two. A full-fledged component, however, will need all of them!

IPipelineComponent

The only required interface for your object will be IPipelineComponent, which defines the entry points used by the pipeline objects to execute the component's code.

IPipelineComponent only has two member functions, Execute and EnableDesign. Although they must both be implemented, only the first one will usually be of any interest; in fact, you can leave EnableDesign empty altogether.

Execute takes a number of parameters that represent the pipeline's entire set of data:

```
IPipelineComponent:Execute (
  Dictionary,
  Context,
  Flags,
  ErrorLevel)
```

The Dictionary and Context parameters contain the pipeline's dictionary and its context respectively. As mentioned earlier, these will be two Dictionary objects—or two objects that implement the IDictionary interface (such as Orderform). Flags is a reserved parameter and should be left empty, while ErrorLevel must be set before exiting the component to a value indicating its result. The possible values are:

1. Success
2. Warning
3. Error

It should be noted that these values have been arbitrarily defined to work with the built-in MSCS pipelines. If you edit the pipeline using the Win32 editor in expert mode, you will be able to set the maximum error level that it can withstand. When the value of ErrorLevel exceeds the maximum value set for a given pipeline, the execution stops and the pipeline returns an error.

The reason why there is a distinction between warning and error conditions is that in the plan pipeline, not all the components may be able to run properly even though the pipeline is being executed with all the appropriate data required for a given step in the purchase process.

For example, suppose the store is at the basket level with a new customer; in that case, no shipping or billing information, which will most likely cause several components in the Shipping, Tax and Handling stages to fail. However, that information *is not needed* in the basket page, and therefore the pipeline should not return an error condition. As a result, the plan pipeline can withstand a maximum error level of two (Warning). The Purchase pipeline, on the other hand, is run only at a critical point in the store, and therefore can only withstand the Success condition.

IPipelineComponentAdmin

When exposing properties for your pipelines, it's always a good idea to implement them as COM properties of your component first. This will allow you to implement any validation that you need for them in one place and one place only, as opposed to having to handle it separately if you implement property bags or property pages. It's also a good idea to make these properties handle Variant values directly, so that they will be easily accessible from ASP scripts. This will mean a little more work if you're using Visual C++, but it will be worth the effort if you are planning to use

Micropipes in your projects or (indirectly) if you want it to be compatible with the HTTP editor.

IPipelineComponentAdmin provides a simple mechanism for setting and retrieving a component's properties through a Dictionary object, which can once again used when you're planning to work with Micropipes or the HTTP editor. The GetConfigData function is called by the object that wants to set your component when it needs to read the configuration parameters that your component will accept:

```
Function IPipelineComponentAdmin.GetConfigData as Object
```

This function should return a Dictionary object that contains a name/value pair for each property that the component supports. The Dictionary should contain a complete collection of all the properties, even if they are Null or empty, because the caller might use GetConfigData to establish how many and which properties the component supports.

SetConfigData is a method of IPipelineComponentAdmin that is called whenever the object's properties must be changed:

```
Function IPipelineComponentAdmin.SetConfigData
(
  dict as Object
)
```

It's important to understand that dict doesn't necessarily have to contain a name/value pair for each and every property that your component exposes. In fact, it could even be empty! This means that your implementation of SetConfigData should enumerate all the pairs in dict and set only the properties that are defined in it. If it doesn't do so and, for example, it simply attempts to read all possible properties and set their value to the value returned by the Dictionary object, it would be unable to determine whether a particular value is Null because the pair doesn't exist in the object or because the caller set it.

IPipelineComponentDescription

The last interface defined by SSCE for writing pipeline components is used to report the values that are read and written by the component to the pipeline's dictionary and context. As you can imagine, there is no strict need to implement this interface, since the information that you return is...well, just information, and it won't be used actively during the pipeline's execution.

However, you should consider that it might be useful for other people to know what the component does when it is run. This is particularly true if you are considering to sell your component as an add-on to SSCE and therefore its users will not be able to contact you directly for more information.

IPipelineComponentDescription provides three methods, all identical except for the nature of the information that they report:

```
Function IPipelineComponentDescription.ContextValuesRead as Variant
```

```
Function IPipelineComponentDescription.ValuesRead as Variant
```

Function IPipelineComponentDescription.ValuesWritten as Variant

ContextValuesRead is used to report the values that the component requires from the pipeline's context (no values should ever be written in the context); similarly, ValuesRead reports the values that the component reads from the pipeline's dictionary, while ValuesWritten reports the values that are written by the component to the dictionary.

All three methods return a Variant value that points to an array of Variants. Each element of the array, in turn, contains a string that corresponds to one of the values that are being reported. If you are working in Visual C++, the array that you return encapsulated in a Variant must of type SAFEARRAY.

Getting ready to write pipeline components

The first step in writing pipeline components consists of creating the appropriate environment, so that you will worry about writing optimal algorithms and code rather than trying to figure out why you can't compile your component.

You should start by making sure that the machine on which you are going to work *knows* what MSCS objects and interfaces are. You won't be able to write a component in Visual Basic, for example, unless all the MSCS objects and interfaces you want to use are registered on that machine.

If your development computer runs on Windows NT Server, you could install a complete copy of Site Server on it—that would ensure that all the required components are registered. If that isn't the case, or if you don't want to install additional copies of SSCE, you will need to copy the DLL files from the folder where an existing copy of SSCE is running.

Next, you'll have to install the Site Server SDK, which is part of the second CD of Site Server. This will be particularly helpful if you're planning to work in Visual C++, since it contains an extension for the ATL wizard that will let you create a new component in just a few simple steps.

Choosing the right language

Because pipeline components are essentially a particular class of COM objects, you will be able to write them using any development tool able to generate apartment- or free-threaded COM objects. This includes Visual Basic and Visual C++, with which most of you will be already familiar, but also other tools, such as Visual J++ and Borland C++ builder. For this presentation, we'll focus on VB and VC++.

Choosing the right language for writing your components is essential to your success. There are at least to aspects of each language that you will want to consider: the overall performance and the convenience. While you will certainly want to lean more on the former for your production environment, convenience will help you concentrate on the validity of your algorithms, rather than on the nuances of a particular language, when you are trying to build a prototype.

If you follow this line of reasoning, VB will certainly be a good choice when you're developing your components and therefore have to first of all make sure that your conceptual design works flawlessly. VB is easy to program, fully compatible with COM and provides a great debugging environment.

You will need to use Visual Basic 5.0 Service Pack 1 or later in order to write pipeline components. Earlier versions of VB do not support the Apartment threading model.

On the other hand, you it's probably fair to say that VB does not provide the best possible performance, not the highest flexibility in what can be done with it. Naturally, you can extend its possibilities by writing DLLs in another language to

which your VB code will interface, but if you have to adopt another language to get things done, why not go all the way and use a development tool that produce faster and more compact code?

If that's what you want to do, Visual C++ will probably be your best choice. While it's certainly more difficult to write a component using this language, the Site Server SDK contains an extension to the project creation wizard that will automatically create the entire skeleton for a complete component with just a couple of mouse clicks.

You will need a special patch to the SDK in order to use the wizard extension with Visual C++ 6.0. You can find it at the following URL: http://support.microsoft.com/support/kb/articles/q214/8/32.asp

Writing pipeline components in Visual Basic

The development of a pipeline component in Visual Basic starts with the creation of a new COM DLL project. You can also add a new custom component to an existing COM DLL project. Keep in mind that the project will have to use the Apartment threading model, or you will not be able to run the pipeline component.

Implementing the basic functionality

Next, you can start by implementing IPipelineComponent. It's a good idea to start by writing an empty implementation for EnableDesign, so that you will be able to forget about it and concentrate on more important matters, such as writing your version of Execute, which, as we have seen earlier on, is required to make your component work properly—or even compile.

The important thing to remember when writing the Execute function is to remember to return a value—if you don't, the pipeline's execution will terminate with an error condition even if your component worked perfectly. Also to keep in mind is the fact that you should never write any value in the context dictionary, because that is discarded when the pipeline has completed its execution.

Finally, be careful about the values that you set in the pipeline's dictionary. As you probably know, pipelines rely on a "trust" system for controlling the flow of operations, rather than on conditional operations. As a result, you should set a value in the dictionary if, and only if, it is Null, which indicates that no other components has already set it.

Handling parameters

A good way to handle configurable parameters, as we mentioned earlier, is to start by implementing them as properties of your component. This will make it possible to centralize all the conversions and validations that you need to perform on each parameter in one place only. It's a good idea to leave the parameters as Variants, which will make it possible to easily use them from ASP scripts.

Once you've implemented your properties, the next logical step will be to implement property pages and property bags; this simple mechanism will make your component easily usable from other COM objects and applications, including the pipeline objects and the Win32 editor. In other words, once you've implemented property bags, your component is ready to go.

Finally, you can implement IPipelineComponentAdmin. This is useful only if you need to make your component compatible with the HTTP editor, since the latter provides a number of built-in functions that make it easier to save and retrieve parameters.

When implementing SetConfigData, keep in mind that the simple fact that a value you extract from the Dictionary object is null alone does not mean that you should set the corresponding property to null as well. In fact, you will receive a null value even if the pair wasn't set at all! Therefore, the best way to retrieve all the correct values is to enumerate the contents of the Dictionary one at a time and set the properties that correspond to what you receive.

Additional steps

To add more user-friendliness to your component, you may want to implement IPipelineComponent. As we have already mentioned, there is no particular requirement to do so, except for the fact that it will make it easier for developers to use your component. There is no need for complex code to write the various methods of IPipelineComponent—all you need to do is return a Variant that contains an array whose dimension corresponds to the number of context or dictionary name/value pairs that you want to report. Each element of the array will, in turn, be a Variant containing a string that describes the name/value pair that you intend to report.

Making the component visible

Your next step should consist of making the component visible to the Win32 and HTTP editors. This is done by adding several GUIDs to the Implemented Categories subkey of the Registry key where your component's information is registered. This not only advertises the component as a pipeline component, but also indicates what stages it has affinity with.

You can start by opening the Registry Editor and look for the component; assuming that you called it MyLibrary.MyComponent, you'll find it under the following key:

HKEY_CLASSES_ROOT/MyLibrary.MyComponent

Within that key, you will find a sub-key called CLSID, whose default value contains the Class ID for the component. If you copy that into the clipboard and then use it to make a search starting from the HKEY_CLASSES_ROOT/CLSID key, you will find the component's main registry entries.

One of the sub-keys in the component's main entry is called Implemented Categories, and contains several other sub-keys that reference a number of Class IDs. This sub-key is used to express what kind of component is described in the Registry, and that's where the editor looks to see whether a COM object is indeed a pipeline component.

In order to be marked as such, a pipeline component needs to implement the following category:

{CF7536D0-43C5-11D0-B85D-00C04FD7A0FA}

Specifying affinities

You will also need to specify the stages to which your component can be added. MSCS provides a different category for each possible stage, plus a special category for those components that work in any stage. They are all listed in the Include\Microsoft Site Server\Commerce\Pipe_Stages.h C++ include file. Even if you are not familiar with VC++, you will not have any problem finding and copying the GUIDs.

Adding support for the HTTP editor

Even after you've added all the necessary entries in the registry, you will not be able to edit your component through

the HTTP editor. This happens, obviously, because the editor has no means to show the property pages for the component.

The solution of this problem consists of creating a set of ASP pages that take care of editing the component's properties. You will need two pages: the first will allow the users to edit the properties, and the second will work as the target of an HTML form and write the new values into the pipeline configuration file. The editor will automatically recognize the two files, as long as their names are generated using the following convention:

Ø The main property page must be called with the same name as the component, with the exception of any dots, which should be turned into underscores. For example, the property page for MyLibrary.MyComponent should be called MyLibrary_MyComponent.asp.

Ø The second page should be called using the same convention as for the main page, with the _Post suffix appended to it (i.e.: MyLibrary_MyComponent_Post.asp).

Writing pipeline components in Visual C++

The essential concepts that we have seen so far also apply to Visual C++ component development. The major difference will be in the increased complexity of the language, which will, however, also result in increased flexibility and performance.

Most of the hurdles of creating a new component and implementing the right interfaces are taken care by the ATL wizard extension that comes with the Site Server SDK. In fact, once you've created a new pipeline component using it, you will end up with a COM object that already implements all the pipeline interfaces, persistence and property pages, as well as providing a plethora of support functions that can be used to perform tasks as varied as accessing the Registry or manipulating MSCS objects.

The helper functions

There are a large number of functions that the template created by the ATL wizard extension provides. They can, however, be grouped in these categories:

Ø **Manipulating MSCS Objects**

These functions make it possible to access MSCS objects, which are usually passed as pointers to IDispatch interfaces. In particular, the functions provided by the template focus on Dictionary and SimpleList components, which will be the most likely candidates for manipulation.

Ø **Generating unique IDs**

The GenUniqueID function generates a unique identifier similar to the ones that an MSCS store uses for order numbers and shopper IDs.

Ø **Accessing error lists and the MessageManager object**

If you need to generate an error and make it available to the pipeline, you can access the MessageManager and the error lists in the Orderform by using these specialized functions.

- **Manipulating VARIANT values**

Since all the values that you extract from a Dictionary object are Variants, it's good to have a few functions that let you convert a VARIANT value into either a string, an integer or an IDispatch pointer, which are easier to handle for a VC++ application.

- **Accessing the Registry**

Finally, if you need to access the Registry while your component is being registered, the template includes several functions that make the process of declaring supported threading models or stages.

Implementing the component's functionality

As before, you will want to start by implementing the Execute method, which is where the heart of your component is. The same considerations that we mentioned above are essentially valid, and the only real difference that you will encounter is in the way the language works when you compare it to Visual Basic. This is the main reason why Visual C++ should be your choice for production but not for prototypes: in addition to the difficulty of having to write a sometimes complex algorithm, you also have to deal with the fact that C++ is a much more complex language (although not necessarily more difficult to use) than VB.

Next, you will want to work your way through the process of implementing properties. Once again, you may find it easier to start by writing the properties as methods of your component, and then adding the property pages and the methods of IPipelineComponentAdmin at a second stage. Keep in mind that working with VARIANT values is not as easy here as it is in VB, although you can use the functions provided by the template to simply convert the Variants into manageable C++ types.

The final step will be that of adding the implementations of the methods that belong to IPipelineComponentDescription. In this case, you will have to create SAFEARRAY objects and return those encapsulate in VARIANT objects. Even though this might be a slightly more difficult task in VC++, the template already contains boilerplate implementations of the methods, so you shouldn't have any problem.

Microsoft Site Server Commerce
Microsoft BackOffice® Family Member

Implementing Pipeline Interfaces in Microsoft Site Server 3.0: Converting Existing COM Components

June 1999

Microsoft Corporation

Summary

This article illustrates how to create a simple component with a simple COM object. Necessary steps are then presented to manually add and implement the interfaces required for a fully functional pipeline component. The scope of this article covers converting COM servers so that they expose pipeline-compliant interfaces.

Introduction

Many existing Component Object Model (COM) objects contain business rules that organizations wish to move into the Internet Commerce arena. This migration can be accomplished in several ways. Very often it is advisable to revisit the business rules and design the components with Internet Commerce architecture in mind. This implies re-architecting the solution from the beginning, bearing in mind what it would mean to add Microsoft® Internet Information Services (IIS), Microsoft Transaction Server (MTS), and Microsoft Site Server Commerce to the execution of the component.

However, sometimes one might wish to modify the component to make it a pipeline-compliant component. Several ways to approach this type of modification include—from external pipeline component wrappers, to component aggregation, to implementing the necessary interfaces in the component to make it pipeline-compatible.

In this article, we create a simple component with a simple COM object. Then we follow the steps necessary to manually add and implement the interfaces required for a fully functional pipeline component. We could have also explored writing a pipeline component that simply used the original COM object, or we could have added a COM object to the original component that encapsulated the original object. However, the scope of this article covers converting COM servers so that they expose pipeline-compliant interfaces.

Most of the information in this document comes directly from the Commerce Documentation and the Commerce SDK. For further information, see http://www.microsoft.com/commerce.

Create the Initial Component

To keep things simple, we will create a simple component to calculate a tax amount:

1. Create a new Active Template Library (ATL) COM AppWizard project, and name it "**SimpleTax**."
2. Create it as a DLL. MTS support is optional.
3. Add a simple ATL Object to the project, and name it "**CityTax**."
4. Right click the **CityTax** interface in the Microsoft Windows Explorer, and add a property named "**Percent**" of type *float*.
5. Right click again on the **CityTax** interface, and add a method named "**CalculateTax**." For parameters, add "**float Amount, float *TotalTax**."
6. Open the SimpleTax.idl file. Look for the idl of the **CalculateTax** method. It should look like this:

   ```
   [id(2), helpstring("method CalculateTax")] HRESULT CalculateTax(float Amount,
   ```

   Modify it to have a return value by adding the following:

   ```
   [id(2), helpstring("method CalculateTax")] HRESULT CalculateTax(float Amount,
   ```

7. Add the following to the **CCityTax** constructor:

   ```
   m_Percent = 0.0;
   ```

8. Add a private member to the **CCityTax** object name m_Percent of type float.

9. Modify the **Percent** property functions for **CCityTax** to be the following:

```
STDMETHODIMP CCityTax::get_Percent(float *pVal)
{
   *pVal = m_Percent;

   return S_OK;
}

STDMETHODIMP CCityTax::put_Percent(float newVal)
{
   m_Percent = newVal;

   return S_OK;
}
```

0. Modify the **CalculateTax** method to look like the following:

```
STDMETHODIMP CCityTax::CalculateTax(float Amount, float *TotalTax)
{
   *TotalTax = Amount * (m_Percent/100);

   return S_OK;
}
```

1. Compile, and the COM DLL should be ready. Test its functionality using your favorite method, i.e., Microsoft® Visual J++®, Microsoft Visual C++®, Microsoft Visual Basic®, Microsoft FoxPro®, Microsoft Visual Basic Scripting Edition (VBScript), and so on.

Converting a COM Component for the Pipeline

Implementing the Mandatory Interface IPipelineComponent

IPipelineComponent is the only **mandatory** interface needed for the component to behave as a pipeline component. It includes two methods: **EnableDesign** and **Execute**. The **Execute** method is where all of the processing takes place, and it is where we will be doing the majority of our work. All other interfaces, though needed, are optional.

1. Copy the Computil.cpp file into your project directory.

2. Add Computil.cpp to your project.

3. Copy the following .H files into the project directory:

Commerce.h	Contains the interface definitions for the Site Server 3.0 Commerce objects.
Comp_ids.h	Contains the class identifiers (CLSIDs) for the pipeline objects and interfaces.
Mspu_guids.h	Contains the CLSIDs for the order processing pipeline objects and interfaces.
Pipeline.h	Contains the definitions for the pipeline interfaces.
Pipecomp.h	Contains the proxy/stub definitions for the pipeline interfaces.
Pipe_stages.h	Contains the GUIDs for the OPP stages.

As an option, you could add the path the SDK includes for Microsoft Site Server 3.0 Commerce Edition (SSCE) to your project by adding a line similar to:

/I "C:\Microsoft Site Server\SiteServer\commerce\sdk\commerce\include" in the C/C++ tab.

4. Add this line to your COM DLL:

   CPP file: #include "mspu_guids.h"

5. Build and resolve any build errors.

6. Add the following to your component's inheritance list:

   public **IpipelineComponent**

7. Add the following to the component's .H file:

```
#include "computil.h"
#include "pipeline.h"
#include "pipecomp.h"
#include "pipe_stages.h"
```

8. Ensure that the declarations for **IPipelineComponent** methods are added to the **CCityTax** declaration. Add the following to the class declaration for your component:

```
// IPipelineComponent
   STDMETHOD(Execute)(
      IDispatch*  pdispOrder,
      IDispatch*  pdispContext,
      LONG        lFlags,
      LONG*       plErrorLevel);
   STDMETHOD (EnableDesign) (BOOL fEnable);
```

9. Implement the methods in the CityTax.cpp file. Add the following to the CityTax.cpp file:

```
// IPipelineComponent Methods
//
STDMETHODIMP CCityTax::Execute (
      IDispatch*  pdispOrder,
      IDispatch*  pdispContext,
      LONG        lFlags,
      LONG*       plErrorLevel)
{
   HRESULT hRes = S_OK;

   // TODO: Add code that performs the main operations for this component
   return hRes;
}

STDMETHODIMP CCityTax::EnableDesign(BOOL fEnable)
{
   return S_OK;
}
```

0. Add the following to the COM Interface map for **CityTax**:

```
   COM_INTERFACE_ENTRY(IPipelineComponent)
```

1. Ensure that, when the component is registered, it is registered for Pipeline stage affinity. Therefore, create a custom registration call and add the following lines to the component's class declaration:

```
static HRESULT WINAPI UpdateRegistry(BOOL bRegister)
{
```

```
    HRESULT hr = _Module.UpdateRegistryClass(GetObjectCLSID(), _T("SimpeTax.Ci
    if (SUCCEEDED(hr))
    {
        // TODO: Add stage affinities here.
        hr = RegisterCATID(GetObjectCLSID(), CATID_MSCSPIPELINE_COMPONENT);
        hr = RegisterCATID(GetObjectCLSID(), CATID_MSCSPIPELINE_ANYSTAGE);
    }
    return hr;
};
```

and comment the following line:

```
DECLARE_REGISTRY_RESOURCEID(IDR_CITYTAX)
```

2. Test our new converted component's functionality in the pipeline, and add some code to the body of the **Execute** method. Later on, you need to modify the functionality of this to use our persisted properties. However, for the moment, hardcode 10%. Also, pay particular attention to the code to manipulate the **Commerce.Dictionary** object. Helper functions for manipulating objects exposed by the Commerce Library can be found in the Computil.H and Computil.CPP files. The code you need to place in the **Execute** method is as follows:

```
IDictionary *pDictOrder = NULL;
HRESULT hr;

VARIANT var;
VARIANT Price;
float Tax;

OPP_ERRORLEV ErrorLevel = OPPERRORLEV_FAIL;
//initialize variants
VariantInit(&var);
VariantInit(&Price);

if(pdispOrder == NULL)
    return E_INVALIDARG;

// Get the OrderForm Dictionary.
if(SUCCEEDED(hr=pdispOrder->QueryInterface(IID_IDictionary, (void**)&pDictOrd
{
    //Test Code
    hr = GetDictValue(pDictOrder, L"Test_Price", &var);
    VariantChangeType(&Price,&var,0,VT_R4);

    //Call the original method for processing.
    m_Percent=.10;  //This will be commented out when  property persistence is
    hr = this->CalculateTax(Price.fltVal , &Tax);

    var.fltVal  = Tax;
    var.vt = VT_R4;
    hr = PutDictValue(pDictOrder, L"Test_Tax", var);
}

if(SUCCEEDED(hr))
    ErrorLevel = OPPERRORLEV_SUCCESS;

if(plErrorLevel) *plErrorLevel = ErrorLevel;

if(pDictOrder)
    pDictOrder->Release();
```

```
return hr;
```

3. Compile the component at this point, and run it in a pipeline or a micropipe. When the component is executed, it will look for a "Test_Price" value in the pdispOrder dictionary, and write a "Test_Tax" value to the same dictionary.

4. Continue developing this component in the rest of this article by implementing the optional interfaces required for configuration. This completes the necessary augmentations to convert the original component into one that can be used by the Pipeline.

Implementing IPipelineComponentDescription

The **IPipelineComponentDescription** is an optional interface that makes it possible for pipeline components to identify the values they read from the pipe context, and the name/value pairs they read or write from or to the **OrderForm**—for order processing pipeline components—or **Dictionary**—for Commerce Interchange Pipeline (CIP) components.

1. Add the following declaration to the **CCityTax** class inheritance list:

```
public IDispatchImpl<IPipelineComponentDescription, &IID_IPipelineComponentDe
```

IDispatchImpl provides a default implementation for the **IDispatch** portion of any dual interface on your object.

2. Add the following to the COM_MAP:

```
COM_INTERFACE_ENTRY(IPipelineComponentDescription)
```

Also, make the following changes in the COM_MAP:

1. Comment out `COM_INTERFACE_ENTRY(IDispatch)`

2. Add `COM_INTERFACE_ENTRY2(IDispatch, ICityTax)`

These modifications are needed, because our object is now derived from two branches of **IDispatch**. Using the COM_INTERFACE_ENTRY2 macro allows us to disambiguate the interfaces.

3. Add the following method declarations to the **CCityTax** class:

```
// IPipelineComponentDescription
STDMETHOD(ContextValuesRead)(VARIANT *pVarRead);
STDMETHOD(ValuesRead)(VARIANT *pVarRead);
STDMETHOD(ValuesWritten)(VARIANT *pVarWritten);
```

4. Give the new methods bodies in the CityTax.cpp file. The code would look like this:

```
//
// IPipelineComponentDescription Methods
//
STDMETHODIMP CCityTax::ContextValuesRead(VARIANT *pVarRead)
{
   // TODO: Add your own values to the array
   int cEntries = 1;
   // allocate the safearray of VARIANTs
   SAFEARRAY*    psa = SafeArrayCreateVector(VT_VARIANT, 0, cEntries);
   // Populate the safearray variants
   VARIANT* pvarT = (VARIANT*)psa->pvData;
   V_BSTR(pvarT) = SysAllocString(L"None");
   V_VT(pvarT) = VT_BSTR;
   // set up the return value to point to the safearray
   V_VT(pVarRead) = VT_ARRAY | VT_VARIANT;
   V_ARRAY(pVarRead) = psa;
   return S_OK;
}
```

```
STDMETHODIMP CCityTax::ValuesRead(VARIANT *pVarRead)
{
   // TODO: Add your own values to the array
   int cEntries = 1;
   // allocate the safearray of VARIANTs
   SAFEARRAY*   psa = SafeArrayCreateVector(VT_VARIANT, 0, cEntries);
   // Populate the safearray variants
   VARIANT* pvarT = (VARIANT*)psa->pvData;
   V_BSTR(pvarT) = SysAllocString(L"Test_Price");
   V_VT(pvarT) = VT_BSTR;
   // set up the return value to point to the safearray
   V_VT(pVarRead) = VT_ARRAY | VT_VARIANT;
   V_ARRAY(pVarRead) = psa;

   return S_OK;
}

STDMETHODIMP CCityTax::ValuesWritten(VARIANT *pVarWritten)
{
   // TODO: Add your own values to the array
   int cEntries = 1;
   // allocate the safearray of VARIANTs
   SAFEARRAY*   psa = SafeArrayCreateVector(VT_VARIANT, 0, cEntries);
   // Populate the safearray variants
   VARIANT* pvarT = (VARIANT*)psa->pvData;
   V_BSTR(pvarT) = SysAllocString(L"Test_Tax");
   V_VT(pvarT) = VT_BSTR;
   // set up the return value to point to the safearray
   V_VT(pVarWritten) = VT_ARRAY | VT_VARIANT;
   V_ARRAY(pVarWritten) = psa;
   return S_OK;
}
```

5. Compile the component. Now if you open your test Pipeline, or create one and add City Tax Class to it, you should be able to look at the properties of the component and see "**Values Read and Written**."

6. This concludes implementation of **IPipelineComponentDescription**.

Implementing ISpecifyPropertyPages

The **ISpecifyPropertyPages** interface is a standard Microsoft Win32® OLE interface. Implement it to allow the pipeline administration tool to invoke the component's property page user interface. For more information about **ISpecifyPropertyPages**, see the *OLE Programmer's Reference*.

1. Add the following declaration to the **CCityTax** class inheritance list:

```
public ISpecifyPropertyPages,
```

2. Add the following to the COM_MAP:

```
COM_INTERFACE_ENTRY(ISpecifyPropertyPages)
```

3. Add the following method declarations to the **CCityTax** class:

```
// ISpecifyPropertyPages
STDMETHOD (GetPages)(CAUUID *pPages);
```

4. Give the **GetPages** method a default implementation in the CityTax.cpp file.

```
if (NULL == pPages)
   return E_INVALIDARG;
// TODO: Uncomment out the line below, and add the CLSID of a custom Property
/*
pPages->cElems = 1;
pPages->pElems = (GUID*)CoTaskMemAlloc(1*sizeof(GUID));
if(!pPages->pElems){
   pPages->cElems = 0;    return E_OUTOFMEMORY;
}
memcpy(pPages->pElems, &CLSID_CityTaxPpg, sizeof(GUID));

return S_OK;
*/
return E_NOTIMPL;
```

5. Create a property page. Right click on the project in the **Windows Explorer ClassView** tab, select **New ATL Object**, choose **Controls**, and then select **Property Page**. Use **CityTaxPPg** for the short name, and click **OK**.

6. Go to the resource view of Windows Explorer, and open the newly created dialog box. Add a static control with the words "**Tax Rate Percentage:**" for the caption.

7. Add an Edit control named "**IDC_Percentage.**"

8. Uncomment the body of the **GetPages** method. It should be correct. If you named your property page something different, you need to change the *CLSID_CityTaxPpg* to reflect the appropriate name.

9. Compile and test in a pipeline. You should get a property page with the static and edit controls that you placed on the dialog box.

0. Make the page communicate with the control. This includes creating a **IPropertyPageImpl::Activate** method and modifying the **IPropertyPageImpl::Apply** method. You need to implement the **Activate** method first.

1. Add "**STDMETHOD(Activate)(HWND hWndParent, LPCRECT pRect, BOOL fModal);**" to the **CCityTaxPpg** class declaration.

2. Create an empty body for the method in the CCityTaxPpg.cpp by adding the following:

```
STDMETHODIMP CCityTaxPpg::Activate(HWND hWndParent, LPCRECT pRect, BOOL fModa
{


   return S_OK;
}
```

3. Obtain a pointer to the component in the **Activate** method so that you can retrieve property values from it. Add the following code to the body of the **Activate** method:

```
USES_CONVERSION;        //this is needed for BSTR conversion macros

VARIANT Percentage;
VARIANT varbstrPercentage;
float Percent;

HRESULT hRes = IPropertyPageImpl<CCityTaxPpg>::Activate(hWndParent, pRect, fM
if(SUCCEEDED(hRes))
{
   ATLTRACE(L"Created CityTax Property Page object");
   CComQIPtr<ICityTax, &IID_ICityTax>pCityTax(m_ppUnk[0]);

   VariantInit(&Percentage);
```

```
    VariantInit(&varbstrPercentage);

    pCityTax->get_Percent(&Percent);

    V_R4(&Percentage) = Percent;
    V_VT(&Percentage) = VT_R4;

    hRes = VariantChangeType(&varbstrPercentage, &Percentage, 0, VT_BSTR);
    ATLTRACE(varbstrPercentage.bstrVal );

    if(FAILED(hRes))
       return hRes;

    SetDlgItemText (IDC_PERCENTAGE, W2A(varbstrPercentage.bstrVal));

    VariantClear(&varbstrPercentage);
    VariantClear(&Percentage);
}
return S_OK;
```

4. Implement the **Apply** method now that **Activate** is complete. In the property page .H file, find the default implementation of **Apply**—it should look like this:

```
STDMETHOD(Apply)(void)
{
    ATLTRACE(_T("CCityTaxPpg::Apply\n"));
    for (UINT i = 0; i < m_nObjects; i++)
    {
       // Do something interesting here
       // ICircCtl* pCirc;
       // m_ppUnk[i]->QueryInterface(IID_ICircCtl, (void**)&pCirc);
       // pCirc->put_Caption(CComBSTR("something special"));
       // pCirc->Release();
    }
    m_bDirty = FALSE;
    return S_OK;
}
```

Replace this in the .H file with:

```
STDMETHOD(Apply)(void);
```

5. Add an implementation of the **Apply** method in the property page .CPP file. The implementation should look like the following:

```
STDMETHODIMP CCityTaxPpg::Apply (void)
{
    ATLTRACE(_T("CCityTaxPpg::Apply\n"));
    CComQIPtr<ICityTax, &IID_ICityTax>pCityTax(m_ppUnk[0]);

    CComBSTR   bstrPercentage;
    VARIANT      varBSTRPercentage, varPercentage;
    HRESULT      hRes;

    VariantInit(&varBSTRPercentage);
    VariantInit(&varPercentage);

    GetDlgItemText(IDC_PERCENTAGE,bstrPercentage.m_str);
```

```
    V_BSTR(&varBSTRPercentage) = SysAllocString(bstrPercentage.m_str);
    V_VT(&varBSTRPercentage) = VT_BSTR;
    hRes = VariantChangeType(&varPercentage,&varBSTRPercentage,0,VT_R4);

    if(FAILED(hRes))
       return hRes;
    pCityTax->put_Percent(varPercentage.fltVal );

    VariantClear(&varBSTRPercentage);
    VariantClear(&varPercentage);

    m_bDirty = FALSE;
    return S_OK;
}
```

6. Ensure that everything you need for your property page is in place. The property page should initialized with the default value and maintain state of the property value. The next phase of this project is to implement property value persistence.

Implementing IPersistStreamInit

The **IPersistStreamInit** interface is a standard Win32 OLE interface. This interface is typically implemented on any object that needs to support initialized stream-based persistence, regardless of whatever else the object does. In the context of the order processing pipeline (OPP), it is used to load/save the component configuration in the pipeline configuration file (the pipeline will call on this interface, as appropriate). For information about **IPersistStreamInit**, see the *OLE Programmer's Reference*.

1. Add **IPersistStreamInit** to the class inheritance list:

```
public IpersistStreamInit,
```

   At this point, your class declaration should like similar to this:

```
class ATL_NO_VTABLE CCityTax :
    public CComObjectRootEx<CComSingleThreadModel>,
    public CComCoClass<CCityTax, &CLSID_CityTax>,
    public IPipelineComponent,
    public ISpecifyPropertyPages,
    public IPersistStreamInit,
    public IDispatchImpl<IPipelineComponentDescription, &IID_IPipelineComponen
    public IDispatchImpl<ICityTax, &IID_ICityTax, &LIBID_SIMPLETAXLib>
{
```

2. Add it to the COM_MAP now, as with the previous interfaces:

```
COM_INTERFACE_ENTRY(IPersistStreamInit)
```

3. Add the method declarations to the **CCityTax** class together with two variables for stream versions:

```
// IPersistStreamInit
    const long  m_lStreamVersionMajor;   // major version number of the stream
    const long  m_lStreamVersionMinor;   // minor version number of the stream
    STDMETHOD(GetClassID)(CLSID *pClassID);
    STDMETHOD(IsDirty)(void);
    STDMETHOD(Load)(IStream *pStm);
    STDMETHOD(Save)(IStream *pStm, BOOL fClearDirty);
    STDMETHOD(GetSizeMax)(ULARGE_INTEGER *pcbSize);
    STDMETHOD(InitNew)(void);
```

4. Add empty implementations for the **IPersistStreamInit** methods to the CityTax.cpp file. The implementations may look like this:

```
STDMETHODIMP CCityTax::IsDirty(void)
{
   return S_OK;
}

STDMETHODIMP CCityTax::Load(IStream *pStm)
{
   HRESULT hRes = S_OK;
   return hRes;
}

STDMETHODIMP CCityTax::Save(IStream *pStm, BOOL fClearDiry)
{
   HRESULT hRes = S_OK;
   return hRes;
}

STDMETHODIMP CCityTax::GetSizeMax(ULARGE_INTEGER *pcbSize)
{
   return S_OK;
}

STDMETHODIMP CCityTax::InitNew(void)
{
   return S_OK;
}

STDMETHODIMP CCityTax::GetClassID(CLSID *pClassID)
{
   *pClassID = GetObjectCLSID();
   return S_OK;
}
```

5. Compile your code; it should compile without errors. If errors are found, take this opportunity to resolve and compile the errors before adding code to the **IPersistStreamInit** methods.

6. Create a local variable. Originally we used a float type to store our value. However, to make things easy in persisting our data, create a local variable of type *CcomVariant,* and assign it with our property value. *CComVariant* implements the **WriteToStream** method, allowing an easy method of persistence. The code that you need to place in the **Save** method would look like this:

```
CComVariant l_Percent;

l_Percent = m_Percent;
hRes = l_Percent.WriteToStream (pStm);
```

7. Do the same for loading the data. Hence, in the **Load** method, add:

```
CComVariant l_Percent;

l_Percent = 0;
l_Percent.ReadFromStream (pStm);

m_Percent = l_Percent.fltVal;
```

8. Implement the **GetMaxSize** method. Add the following:

```
CComVariant l_Percent;

l_Percent = m_Percent;

pcbSize->LowPart = sizeof(l_Percent);
pcbSize->HighPart = 0;
```

9. Go back to the **Execute** method, and modify the code to use the original method for calculation:

```
//Call the original method for processing
//m_Percent=.10;
hr = CalculateTax(Price.fltVal , &Tax);  //m_Percent * Price;
```

0. Compile your code. At this point you should have a fully functional Commerce Pipeline component—the only caveat being that you will want to remove the ATL_MIN_CRT flag from your compile options.

Adding MTS Support

In addition to exporting the mandatory and optional interfaces that the pipeline component is expected to have, we should support the Microsoft Transaction Server. With the release of SSCE 3.0, the MTSTxPipeline was introduced. This pipeline implementation is transactional. SSCE still supports the original non-transacted pipeline, but it runs inside MTS. Often it is desired, if not necessary, that custom pipeline components participate in distributed transactions. The following instructions cover what is needed to add MTS support to our pipeline component.

1. Add the following code to the CityTax.h file:

```
#include "mtx.h"
```

2. Declare a variable for the object context:

```
IObjectContext *Context = NULL;
```

3. Get the executing object transaction context. Hence, add the following call to the top of the **CCityTax::Execute** method:

```
//Get MTS Context
hr=GetObjectContext(&Context);
```

4. Handle the response that you receive after the call to **GetObjectContext**. The following is a skeletal switch statement you might implement immediately after the **GetObjectContext** call:

```
//Handle MTS respone
switch(hr)
{
case S_OK:
   break;
case E_INVALIDARG:
   break;
case E_UNEXPECTED:
   break;
case CONTEXT_E_NOCONTEXT:
   break;
}
```

5. Ensure that the additions are complete. Call the **SetAbort** method if anything fails and the **MTS ObjectContext** is valid. Otherwise, if the **ObjectContext** is valid and nothing fails, call **SetComplete**. The following code is an example of what you might add to the end of the **CCityTax::Execute** method implementation:

```
if(SUCCEEDED(hr))
{
```

```
    ErrorLevel = OPPERRORLEV_SUCCESS;
    if(Context != NULL){hr=Context->SetComplete ();}
}
else
{
    if(Context != NULL){hr=Context->SetAbort ();}
}
```

6. Ensure that changes necessary for adding MTS support are complete. It is necessary to register the component in MTS. Do not register the part of the COM server that serves up the property pages. One last caveat—if you wish to create other objects inside the current transaction context, use the following call:

```
//MTS:  How to create an object within this transaction context.
hr = Context->CreateInstance(CLSID_MyID, IID_IMyObjInterface, (void**)&pMyObj
```

U. S. DEPARTMENT OF COMMERCE
PATENT AND TRADEMARK OFFICE
WASHINGTON, DC 20231
IF UNDELIVERABLE RETURN IN TEN DAYS

OFFICIAL BUSINESS

AN EQUAL OPPORTUNITY

US 0994428708P1

Creation date: 10-27-2004
Indexing Officer: SKHUON - SREYTO KHUON
Team: 3600PrintWorkingFolder
Dossier: 09944287

Legal Date: 10-27-2004

No.	Doccode	Number of pages
1	RETMAIL	7

Total number of pages: 7

Remarks:

Order of re-scan issued on .....